TOIMINTA JA HALLINTO 2007:25

OpenDocument-standardi asiakirjojen tallennusmuotona

ISSN 1458-6436
ISBN 978-952-466-598-8 (nid.)
ISBN 978-952-466-599-5 (PDF)
Oikeusministeriö
Helsinki

OIKEUSMINISTERIÖN TOIMINTA JA HALLINTO 2007:25

OpenDocument-standardi asiakirjojen tallennusmuotona

OIKEUSMINISTERIÖ
HELSINKI 2007

KUVAILULEHTI

O I K E U S M I N I S T E R I Ö

Julkaisun päivämäärä
20.8.2007

Tekijät (toimielimestä: toimielimen nimi, puheenjohtaja, sihteeri) Marko Grönroos Martti Karjalainen	Julkaisun laji Selvitys Toimeksiantaja Oikeusministeriö Toimielimen asettamispäivä
Julkaisun nimi OpenDocument-standardi asiakirjojen tallennusmuotona	
Julkaisun osat	

Tiivistelmä

OpenDocument on uusi XML-rakenteeseen pohjautuva tiedostomuoto tyypillisille toimistosovellusten käsittelemille asiakirjoille, kuten tekstiasiakirjoille, laskentataulukoille ja graafisille esityksille. XML-rakenne luo edellytykset asiakirjoja käsittelevien järjestelmien yhteentoimivuudelle ja asiakirjojen sisältämien tietojen jatkokäsittelyn automatisoinnille. OASIS-yhteistyöjärjestön kehittämä avoin OpenDocument-tiedostomuoto hyväksyttiin ISO-standardiksi ISO/IEC 26300 vuonna 2006.

Oikeusministeriö teki vuonna 2006 päätöksen siirtymisestä OpenOffice.org-ohjelmiston ja OpenDocument-tiedostomuodon käyttöön. Käyttöönotto on alkanut vuoden 2007 alusta lukien.

OpenDocument-raportti on laadittu oikeusministeriön OpenOffice- ja OpenDocument-käyttöönoton yhteydessä suomenkieliseksi perusinformaatioksi tiedostomuodosta ja sen hyväksikäytön tietoteknisistä perusteista. Raportissa on kuvattu yleisellä tasolla OpenDocument-tiedostomuoto, sen rakenne sekä kehittämisen ja standardoinnin tausta. Raportissa on myös yhteenveto tiedostomuodon käyttöönottotilanteesta ja avointen tiedostomuotojen EU-tasoisista suosituksista.

Raportin teknisessä osuudessa on kuvattu tarkemmin tiedostomuodon rakenne eri asiakirjatyypeille. Tiedostomuodon hyväksikäyttöä on kuvattu useilla käyttötapauksilla ja niiden teknisen toteutuksen periaatteilla.

Avainsanat: (asiasanat) Toimisto-ohjelmat, OpenOffice.org, OpenDocument, ODF			
Muut tiedot (Oskari- ja HARE-numero, muu viitenumero) OM 15/042/2005			
Sarjan nimi ja numero Oikeusministeriön toiminta ja hallinto 2007:25		ISSN 1458-6436	ISBN 978-952-466-598-8 (nid.) 978-952-466-599-5 (PDF)
Kokonaissivumäärä 80	Kieli suomi	Hinta	Luottamuksellisuus julkinen
Jakaja Oikeusministeriö		Kustantaja Oikeusministeriö	

Sisällysluettelo

1. JOHDANTO

OpenDocument tai lyhyemmin ODF on uusi XML-rakenteeseen pohjautuva tiedostomuoto tyypillisille toimistosovellusten käsittelemille asiakirjoille, kuten tekstiasiakirjoille, laskentataulukoille ja graafisille esityksille. XML-rakenne luo edellytykset asiakirjoja käsittelevien järjestelmien yhteentoimivuudelle ja asiakirjojen sisältämien tietojen jatkokäsittelyn automatisoinnille. OASIS-yhteistyöjärjestön kehittämä avoin OpenDocument-tiedostomuoto hyväksyttiin ISO-standardiksi ISO/IEC 26300 vuonna 2006. Kansainvälisesti hyväksytty avoin standardi vapauttaa asiakirjat sidoksista ohjelmiin, joilla asiakirjoja muodostetaan ja käsitellään, mikä puolestaan mahdollistaa tietojärjestelmä- ja sovellushankintojen aidon kilpailuttamisen ja estää lukkiutumista yksittäisten ohjelmistovalmistajien järjestelmäratkaisuihin.

OpenDocument on uutuudestaan huolimatta jo useiden toimisto-ohjelmien tukema tiedostomuoto. Vuoden 2005 lopulla julkistettu avoimen lähdekoodin OpenOffice.org versio 2 käyttää oletustallennusmuotonaan OpenDocument-tiedostomuotoa. Myös avoimen lähdekoodin KOffice tukee OpenDocument-tiedostomuotoa, kuten myös kaupallisiin ohjelmistoihin kuuluvat Sun Microsystemsin StarOffice ja IBM:n Workplace.

Oikeusministeriö teki vuonna 2006 päätöksen siirtymisestä OpenOffice.org-ohjelmiston ja OpenDocument-tiedostomuodon käyttöön. Käyttöönotto on alkanut vuoden 2007 alusta lukien, ja se on laajuudeltaan toistaiseksi Suomen suurin käsittäen yli 10 000 työaseman tietoteknisen ympäristön.

OpenDocument-raportti on laadittu oikeusministeriön OpenOffice- ja OpenDocument-käyttöönoton yhteydessä suomenkieliseksi perusinformaatioksi tiedostomuodosta ja sen hyväksikäytön tietoteknisistä perusteista. Raportissa kuvataan yleisellä tasolla OpenDocument-tiedostomuoto, sen rakenne ja kehittämisen tausta. Raportissa on myös yhteenveto tiedostomuodon käyttöönottotilanteesta ja avointen tiedostomuotojen EU-tasoisista suosituksista. Tiedostomuodon käyttömahdollisuuksia tuodaan esille vienti- ja tuontisuotimien sekä XForms-lomakkeiden periaatteiden esittelyllä. Luvut 1-4 käsittelevät OpenDocument-muotoa ja sen käyttöä yleisellä tasolla. Luvussa 5 esitetään konkreettisia käyttötapauksia tietojärjestelmäsuunnittelun näkökulmasta. Luvut 6-11 tarjoavat teknisen johdannon OpenDocument-muodosta sekä sen käsittelystä eri työkaluilla. Luvussa 12 annetaan yleiskatsaus XForms-lomakkeista. Raportin yhteenveto esitetään luvussa 13. Raportin lopussa olevissa liitteissä annetaan lisäksi yleiskuvaus joistain erillisistä teknisistä aiheista, kuten XML-merkintäkielestä, XSLT-muunnoskielestä ja DTD- ja RelaxNG-rakennemäärittelyistä.

Raportin ovat laatineet Marko Grönroos ja Martti Karjalainen.

 Raportin PDF-muotoinen verkkoversio on oikeusministeriön verkkosivuilla osoitteessa

http://www.om.fi/Etusivu/Julkaisut/Toimintajahallinto/Toiminnanjahallinnonarkisto/Toimintajahallinto2007

2. AVOIN TIEDOSTOMUOTO

Toimisto-ohjelmilla tuotettujen tekstinkäsittelyasiakirjojen, laskentataulukoiden ja graafisten esitysten tiedostomuodot ovat käyneet läpi jo yli 25 vuoden kehitysjakson. Alkuvaiheessa 1980-luvulla oli käytössä runsaasti keskenään kilpailevien valmistajien ohjelmia, kuten WordStar, WordPerfect, MultiMate, Microsoft Word DOS, IBM DisplayWriter, Visicalc, Lotus 1-2-3 ja Microsoft Multiplan. Eri ohjelmien käyttämät tiedostomuodot olivat keskenään huonosti yhteensopivia, mutta koska ohjelmia käytettiin pääasiassa henkilökohtaisen työskentelyn apuvälineinä ilman liitoksia organisaatioiden tietojärjestelmiin, epäyhteensopivuutta ei koettu vakavaksi ongelmaksi.

Toimisto-ohjelmat koostettiin 1990-luvuilla paketeiksi, ja markkinat kaventuivat muutamaan tuotteeseen. Merkittävimpiä olivat Microsoft Office, IBM Lotus SmartSuite, WordPerfect Office (Borland Office) ja kotikäyttöön tarkoitettuihin tietokoneisiin usein jo valmiiksi asennettu Microsoft Works. Microsoft Office saavutti 1990-luvulla valta-aseman, niin että Office-pakettiin kuuluvien ohjelmien (Word, Excel, PowerPoint) käyttämistä binäärisistä doc-, xls- ja ppt-tiedostomuodoista muodostui 1990-luvun loppuun mennessä dominoiva asiakirjojen tallennuksen ja vaihdon tiedostomuoto.

Sekä Microsoft Officen, IBM Lotus SmartSuiten että WordPerfect Officen käyttämät tiedostomuodot ovat omistusoikeudellisesti rajattuja ja suljettuja, ja tiedostomuotojen täydellinen määritys ja käyttöoikeus on ohjelmistoyritysten hallussa ja valvonnassa.

Asiakirjojen käytössä on siirrytty tilanteeseen, jossa henkilökohtaisten ja yhdellä työasemalla käytettävien asiakirjojen sijaan niitä vaihdetaan organisaatioiden kesken ja niiden tiedot halutaan liittää organisaatioiden tietojärjestelmiin. Suljetut tiedostomuodot ovat asiakirjojen ja niitä käsittelevien järjestelmien yhteentoimivuuden ja pitkäaikaissäilytyksen kannalta kestämättömiä. Lisäksi ne johtavat helposti tiedostomuotoihin perustuviin sidoksiin, toimisto-ohjelmien pakolliseen päivityskierteeseen ja tilanteisiin, joissa tietojärjestelmähankintojen aito kilpailuttaminen ei ole mahdollista. Erityisesti EU:n piirissä onkin oltu aktiivisia tilanteen korjaamiseksi ja avoimien standardien kehittämiseksi asiakirjojen tiedostomuodoille.

Avoimien tiedostomuotojen käytöllä voidaan välttää suljettujen tiedostomuotojen haitat ja saavuttaa mm. seuraavia etuja:

- Avoin tiedostomuoto luo edellytykset eri ohjelmistojen yhteentoimivuudelle asiakirjojen välityksessä ja sähköisten palvelujen kehittämisessä.
- Avoin tiedostomuoto luo edellytykset vuorovaikutteisten ja automatisoitujen prosessien kehittämiseen asiakirjojen käsittelyssä. Esimerkkeinä voidaan mainita sovellusintegrointi, tietojen haku ja raportointi sekä asiakirjojen kokoaminen ja niiden automaattiset muunnokset.
- Avoin tiedostomuoto luo edellytykset asiakirjojen pitkäaikaiseen säilyttämiseen. Vanhojen asiakirjojen avaamiseen voidaan tarvittaessa laatia uudet ohjelmat, jos asiakirjojen laadinnan alkuperäiset ohjelmat ovat poistuneet käytöstä.
- Avoin tiedostomuoto vapauttaa asiakirjat sidoksista ohjelmiin, joilla asiakirjoja muodostetaan ja käsitellään. Tiedostomuodon ja ohjelmien välisen sidoksen purkaminen estää samalla lukkiutumisen yksittäisten ohjelmistovalmistajien järjestelmäratkaisuihin.
- Avoin tiedostomuoto asettaa kaikki ohjelmisto- ja palvelutoimittajat

yhdenvertaiseen asemaan, mikä luo edellytykset tietojärjestelmä- ja sovellushankintojen aitoon kilpailuttamiseen.

Termille "avoin" tiedostomuodon yhteydessä voidaan antaa useita tulkintoja käyttötarkoituksesta riippuen. Avoimuus voi tarkoittaa esimerkiksi seuraavia perusvaatimuksia tiedostomuodolle, sen määrittelylle ja edelleen kehittämiselle:

- Avoin tiedostomuoto on kansainvälisen standardointijärjestön standardoima ja sen dokumentaatio on kenen tahansa saatavilla ja käytettävissä kohtuullisilla kustannuksilla.
- Tiedostomuoto on vapaasti ja veloituksetta käytettävissä ilman omistusoikeudellisia rajoituksia käyttäjien omissa sovelluksissa tai tuotteissa.
- Tiedostomuoto kehitetään yhteistyönä avoimessa prosessissa, johon asiasta kiinnostuneet osapuolet, kuten ohjelmistovalmistajat, keskeiset käyttäjäryhmät ja riippumattomat asiantuntijat, voivat osallistua.

Mahdolliset avoimuuden lisävaatimukset voivat koskea esimerkiksi riippumattomuutta käyttöjärjestelmäalustasta. Seuraavassa on esimerkkejä tunnetuista avoimista tiedostomuodoista.

- ODF (OpenDocument, Open Document Format for Office Applications) on kansainvälisen standardin ISO/IEC 26300 määrittelemä tiedostomuoto muokkauskelpoisille asiakirjoille.
- HTML (HyperText Markup Language) on kansainvälisen standardin ISO/IEC 15445 määrittelemä tiedostomuoto verkkosivuille. HTML on muokkauskelpoinen, mutta ei säilytä tarkasti muotoiluja.
- Raakatekstiesitys on ISO Latin 1 -merkistöön pohjautuva kansainvälisen standardin ISO/IEC 8859-1 määrittelemä tekstin esitysmuoto. Raakateksti on muokkauskelpoinen, mutta ei sisällä muotoiluja.

Adoben kehittämä PDF/A (Portable Document Format Archive) ei tarkkaan ottaen täytä kaikkia edellä esitettyjä vaatimuksia, koska sen kehittäminen on yhden kaupallisen valmistajan (Adobe) valvonnassa. Muilta osin PDF/A on avoin, kansainvälisen standardin ISO/IEC 19005-1 määrittelemä tiedostomuoto ei-muokattaville asiakirjoille erityisesti arkistoinnin tarpeisiin.

3. OPENDOCUMENT-STANDARDIN KEHITTYMINEN

Asiakirjojen standardoitu tiedostomuoto ei ole mikään uusi keksintö, sillä jo 1980-luvulla aloitettiin avoimen asiakirja-arkkitehtuurin (ODA, Open Document Architecture) määrittely. Työ eteni hitaasti, mutta sen tuloksena julkaistiin lopulta vuonna 1999 kansainvälinen standardi ISO 8613.

ODA:n tarkoituksena oli mahdollistaa tekstiä ja kuvia sisältävien asiakirjojen esittäminen ja käsittely ilman, että oltaisiin riippuvaisia valmistajakohtaisista tiedostomuodoista tai laitteista. Tavoitteissa kuitenkin epäonnistuttiin, sillä ODA oli liian monimutkainen ja kunnianhimoinen sen ajan tekniikalle. ODA-standardia tukevia välineitä ei tullut käyttöön.

Avointen tiedostomuotojen kehittäminen sai uutta vauhtia, kun EU:n toimesta julkaistiin vuonna 2004 tulokset avointen tiedostomuotojen selvitystyöstä (ns. Valoris-raportti [Eu2004a]) ja laadittiin suositukset tiedostomuotojen käyttöön julkishallinnossa (ns. TAC-suositukset [Eu2004b]). Suosituksilla pyrittiin avoimien ja standardoitujen XML-pohjaisten tiedostomuotojen käyttöön ja välttämään omistusoikeudellisia tiedostomuotoja. Suosituksiin sisältyi periaate, että julkisen hallinnon ei tulisi asettaa sidosryhmilleen tiedostomuodon kautta pakkoa tietyn tuotteen käyttämiseen.

Avoimeen lähdekoodiin pohjautuvan OpenOffice.org -ohjelmiston käyttämä XML-pohjainen tiedostomuoto luovutettiin EU:n TAC-suositusten mukaisesti OASIS-järjestölle toimisto-ohjelmia koskevan avoimen standardin pohjaksi (OASIS = Organization for the Advancement of Structured Information Standards). OASIS julkisti toukokuussa 2005 toimisto-ohjelmien tiedostomuodoksi tarkoitetun OpenDocument-standardin (Open Document Format for Office Applications v1.0) [Odf2005]. Standardi on veloituksetta kenen tahansa käytettävissä ja sen määritykset ovat julkisesti saatavilla seuraavassa verkko-osoitteessa:

http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=office

Standardi läpäisi äänestyksen ISO-standardiksi keväällä 2006 [Iso2006]. Standardi ISO/IEC 26300 määrittelee tiedostomuodot toimistosovellusten tyypillisille asiakirjoille kuten tekstiasiakirjoille, laskentataulukoille ja graafisille esityksille. Standardi julkaistiin virallisesti 30.11.2006.

OASIS on kehittänyt OpenDocument-tiedostomuotoa edelleen. Vuoden 2007 alussa hyväksyttiin OASIS-standardista versio 1.1, jossa tiedostomuotoon on lisätty vammaisten henkilöiden tietokonekäyttöä tukevia lisäpiirteitä. Vuoden 2007 aikana on vielä odotettavissa versio 1.2, jossa vammaiskäytön tukea laajennetaan edelleen ja täsmennetään lisäksi laskentataulukoiden kaavojen esitystä.

3.1. OpenDocument-standardin käyttöönottotietoja

Jo ennen virallista valmistumistaan OpenDocument-standardi herätti laajaa mielenkiintoa. Se oli ensimmäinen standardoitu asiakirjojen XML-pohjainen tiedostomuoto, joka kattoi tekstiaineiston lisäksi myös taulukot ja graafiset esitykset. XML-rakenteen ansiosta tiedostomuodon katsottiin parantavan oleellisesti asiakirjoja käsittelevien järjestelmien yhteentoimivuutta ja luovan edellytyksiä asiakirjojen sisältämien tietojen jatkokäsittelyn automatisoinnille.

Standardi on saanut hyvän vastaanoton, ja se on jo useiden toimisto-ohjelmien tukema tiedostomuoto. OpenOffice.org versio 2 julkistettiin 20. lokakuuta 2005, ja se käyttää oletustiedostomuotonaan OpenDocument-tiedostomuotoa. Muista toimisto-ohjelmista KOffice ja Sun Microsystemsin StarOffice tukevat OpenDocument-standardia, kuten myös IBM:n Workplace. IBM:n vuonna 2006 julkistaman tiedon mukaan Notes/Dominon uusi versio 8 tukee OpenDocument-tiedostomuotoa. Myös Corel ilmoitti vuonna 2006 kehittävänsä WordPerfect Officeen OpenDocument-tukea, joka tulee saataville vuonna 2007. Microsoft ilmoitti vuonna 2006 rahoittavansa OpenDocument-tuen kehittämistä Microsoft Officeen avoimen lähdekoodin SourceForgen kautta. Vuoden 2007 alussa julkistettiin tämän kehitystyön tuloksena OpenDocument-tuki Microsoft Wordiin. Tuki Exceliin ja PowerPointiin on odotettavissa vuoden 2007 aikana. Microsoftin lisäksi myös useat muut organisaatiot (mm. Sun Microsystems) kehittävät OpenDocument-tukea Microsoft Officeen.

Tarkempia tietoja OpenDocument-tiedostomuotoa tukevista ohjelmista on Wikipediassa seuraavassa verkko-osoitteessa:

http://en.wikipedia.org/wiki/OpenDocument_software

OpenDocument-tiedostomuoto on saavuttanut nopeasti jalansijaa käyttäjäorganisaatioissa. Julkisuudessa on ollut eniten USA:ssa Massachusettsin osavaltio, joka julkaisi syyskuussa 2005 uuden kokonaisarkkitehtuurin ja sen tiedostomuotoja koskevat linjaukset [Mas2005]. Siinä oli hyväksytty uutta arkkitehtuuria noudattaviksi avoimiksi tiedostomuodoiksi seuraavat:

- OpenDocument -tiedostomuoto (teksti- ja taulukkoasiakirjat, graafiset esitykset)
- ASCII-raakatekstimuoto (muotoilematon teksti)
- HTML (selaimella käytettävät asiakirjat)
- PDF (valmiiksi muotoillut asiakirjat, joita ei enää muokata)

EU-valtioista Belgia on tehnyt samansuuntaisen linjauksen kuin Massachusettsin osavaltio. Heinäkuussa 2006 Belgiassa tuli voimaan päätös, jonka mukaan syyskuusta 2008 alkaen hallinnon organisaatioiden välisessä asiakirjavaihdossa on käytettävä OpenDocument-tiedostomuotoa [Bel2006]. Kesäkuussa 2006 Tanskan eduskunta teki päätöksen, jonka mukaan julkishallinnon tietotekniikkaratkaisuissa on siirryttävä 1.1.2008 mennessä avoimiin standardeihin [Den2006]. Päätöksen seurauksena Tanskan tiede- ja teknologiaministeriön verkkosivuilla aineistot ovat olleet saatavilla OpenDocument-muodossa 1.9.2006 alkaen. Myös Ranskassa, Espanjassa, Italiassa ja Norjassa on tehty OpenDocument-muotoa tukevia kansallisia linjauksia ja julkistettu hankkeita avoimiin tiedostomuotoihin siirtymiseksi.

Ranskassa on raportoitu viime vuosien aikana useita suurten organisaatioiden OpenDocument-käyttöönottoja. Heinäkuussa 2006 julkaistun uutisen mukaan Ranskan keskushallinnossa otetaan käyttöön OpenDocument-tiedostomuoto ja OpenOffice-ohjelmisto kaikkiaan 400 000 työasemassa [Fre2006]. Käyttöönoton aikataulun takarajana on vuosi 2007. Siirtymä on valmisteltu vuoden 2006 alussa perustetussa valtion modernisointiyksikössä (DGME, Directorate General for the Modernisation of the State). DGME on valmistellut siirtymää tuottamalla tarvittavat tukivälineet OpenDocument- ja OpenOffice-käyttöönottoon.

Suomen ensimmäiset OpenDocument-käyttöönotot olivat lähinnä OpenOffice-ohjelmiston käyttöönottoja, koska käyttöönotot oli jo tehty ennen tiedostomuodon standardointia.

Ensimmäisiä OpenOfficeen siirtyneitä olivat Suomen perus- ja lähihoitajaliitto SuPer ja Invalidiliitto, joissa siirtyminen tapahtui vuosien 2003-2004 aikana. Muita aikaisin aloittaneita olivat Lemin kunta ja Kokkolan seurakuntayhtymä. Oikeusministeriö teki vuosina 2005-2006 laajan selvitys- ja pilotointivaiheen, jonka tuloksena oikeusministeriö päätti joulukuussa 2006 ottaa käyttöön OpenOffice-ohjelmiston ja OpenDocument-tiedostomuodon vuoden 2007 alusta lähtien (ks. [Fin2006] ja [Fin2007a]). Käyttöönotto on toistaiseksi Suomen julkishallinnon suurin. OpenDocument-tiedostomuoto oli merkittävä peruste myös vuoden 2007 maaliskuussa julkistetussa Uudenkaupungin OpenOffice-käyttöönotossa [Fin2007b].

OpenDocument-tiedostomuodon käyttöönotoista on Wikipedia-sivu seuraavassa verkko-osoitteessa:

http://en.wikipedia.org/wiki/OpenDocument_adoption

4. EU-SUOSITUKSET AVOIMISTA TIEDOSTOMUODOISTA

EU:lla on ollut aktiivinen rooli avoimien tiedostomuotojen kehittymisessä. Euroopan komission alainen IDABC (Interoperable Delivery of European eGovernment Services to public Administrations, Business and Citizens) kehittää julkishallinnon organisaatioiden välisen sähköisen tiedonvaihdon edellytyksiä sekä kansalaisille ja yrityksille suunnattujen palvelujen yhteentoimivuutta. IDABC:n toiminta käynnistyi vuoden 2004 jälkipuoliskolla. Siihen saakka vastaavaa kehitystyötä teki IDA (Interchange of Data between Administrations).

IDABC:n edeltäjä IDA julkaisi huhtikuussa 2004 toimisto-ohjelmia ja niissä käytettyjä tiedostomuotoja koskevan selvityksen, ns. Valoris-raportin [Eu2004a]. Raportissa tarkastellaan toimisto-ohjelmien markkinatilannetta ja erityisesti eri ohjelmien käyttämiä tiedostomuotoja. Tarkastelun tavoitteena oli luoda pohja EU:n laajuisille julkishallintoa koskeville suosituksille toimisto-ohjelmien tiedostomuodoista dokumenttien yhteiskäytön mahdollistamiseksi.

Valoris-raportin perusteella EU:ssa laadittiin vuonna 2004 suositukset avointen tiedostomuotojen käyttöön julkishallinnossa. TAC:n (Telematics between Administrations Committee) hyväksymät ja IDABC:n julkaisemat suositukset on kuvattu tarkemmin lähteessä [Eu2004b]. EU:n sähköisen hallinnon yleinen yhteentoimivuuskehys (European Interoperability Framework, EIF) on esitetty julkaisussa [Eu2004c].

EU:n jäsenvaltioiden hallinnossa tulisi noudattaa yhteentoimivuuskehyksen EIF:n suosituksia [Eu2004c] palvelujen kehittämisessä, jotta yhteentoimivuus EU:n tasolla toteutuisi. Suositusten yleisperiaatteisiin kuuluu mm. avoimien standardien käyttäminen ja avoimeen lähdekoodiin perustuvien ratkaisujen etujen huomioon ottaminen ratkaisuvaihtoehtojen arvioinnissa. Avoimilta standardeilta edellytetään, että ne on laadittu ja niitä kehitetään ei-kaupallisten organisaatioiden toimesta ja että standardit ovat vapaasti saatavilla ja käytettävissä ilman tekijänoikeuskorvauksia.

IDABC:n tiedostomuotoja koskevien ns. TAC-suositusten [Eu2004b] ytimenä on avoimien, XML-pohjaisten tiedostomuotojen käyttö julkishallinnossa. Suljettuja, omistusoikeudellisia tiedostomuotoja tulee suositusten mukaan välttää, ja julkishallinnon tulee suositusten mukaan tarjota asiakirjat sidosryhmille mahdollisuuksien mukaan useissa vaihtoehtoisissa muodoissa.

Edellä kuvatut TAC-suositukset ovat peräisin vuodelta 2004. Suositusten uusin versio on joulukuulta 2006 (ns. PEGSCO-suositukset [Eu2006]), ja niiden keskeinen sisältö julkishallinnolle voidaan tiivistää seuraavasti:

- Julkishallinnon suositellaan käyttävän maksimaalisesti hyväkseen kansainvälisesti standardoituja avoimia tiedostomuotoja sekä sisäisesti että tiedonvaihdossa muiden kanssa.
- Julkishallinnon suositellaan käyttävän vain tiedostomuotoja, jotka ovat monien ohjelmatuotteiden tukemia ja joiden avulla vältetään sidosryhmiltä tietyn tuotteen käyttöpakko. Jos on välttämätöntä käyttää omistusoikeudellista suljettua tiedostomuotoa, tulisi tarjota vaihtoehtona myös kansainvälisesti standardoitu avoin tiedostomuoto.
- Julkishallinnon suositellaan laativan kansallisia linjauksia avoimista

tiedostomuodoista kansainvälisten standardien pohjalta. Linjauksissa tulisi huomioida sekä muokattavien että ei-muokattavien asiakirjojen käyttötarkoitukset.

- Julkishallinnon suositellaan määrittelevän tietojärjestelmien yhteentoimivuuden kannalta keskeiset vähimmäisvaatimukset avoimille tiedostomuodoille.

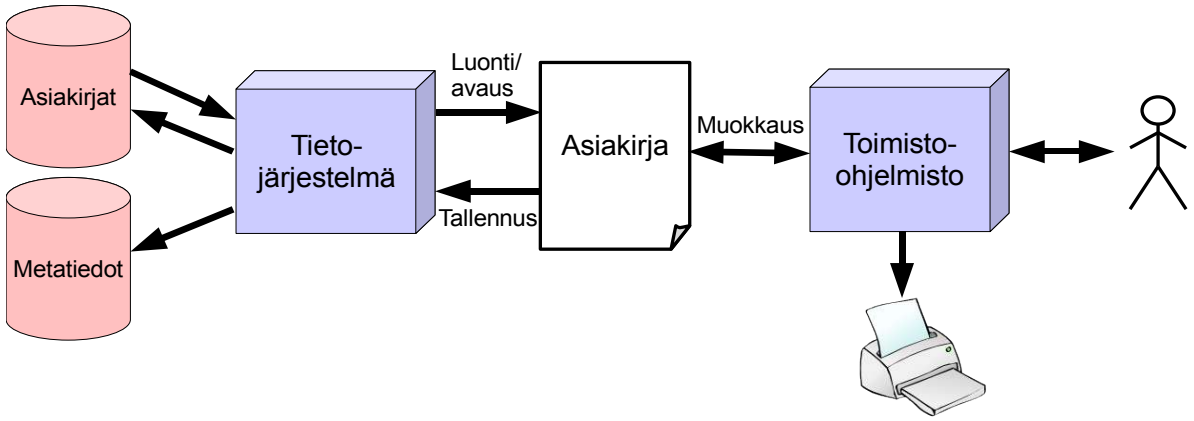
5. KÄYTTÖTAPAUKSIA JA RATKAISUJA

OpenDocument-standardilla on kasvava merkitys asiakirjojen hallinnassa ja sitä toteuttavien tietojärjestelmien integroinnissa toimisto-ohjelmistoihin. Standardi tiedostomuoto, joka on useiden ohjelmistotoimittajien tukema, on tällaisen integroinnin peruslähtökohta. Tässä luvussa esitetään erilaisia OpenDocument-asiakirjojen käyttötapauksia ja ratkaisuja.

Sikäli kun käyttötapaukset koskevat toimisto-ohjelmistoa, keskitymme OpenOfficeen, mutta myös muut OpenDocument-tiedostomuotoa tukevat ohjelmistot tarjoavat vastaavia mahdollisuuksia.

5.1. Toimisto-ohjelmisto osana työnkulkua

Tekstinkäsittely- ja taulukkolaskentaohjelmat ovat keskeinen osa useimpien organisaatioiden toimintaprosessiin liittyvää työnkulkua ja sitä toteuttavaa tietojärjestelmää. Ne mahdollistavat joustavuutensa vuoksi paitsi tavanomaisen kirjallisen tuotannon, myös sovellusaluekohtaisten tietojärjestelmien tai niiden toiminnallisuuksien puutteiden paikkaamisen siltä osin, kun tarvittavia toiminnallisuuksia ei vielä ole saatavilla erikoistuneen järjestelmän muodossa. Tyypillisenä esimerkkinä tästä voidaan ottaa vaikkapa pienimuotoisen postituksen osoitelistojen tekeminen taulukkolaskentaohjelmalla varsinaisen henkilörekisterijärjestelmän käytön sijaan. Tässä käsiteltävän ohjelmistointegraation kannalta oleellisia tilanteita ovat sellaiset, joissa toimisto-ohjelmiston käyttö tiettyjen toimintojen tekemiseen on suunniteltu osa järjestelmää.

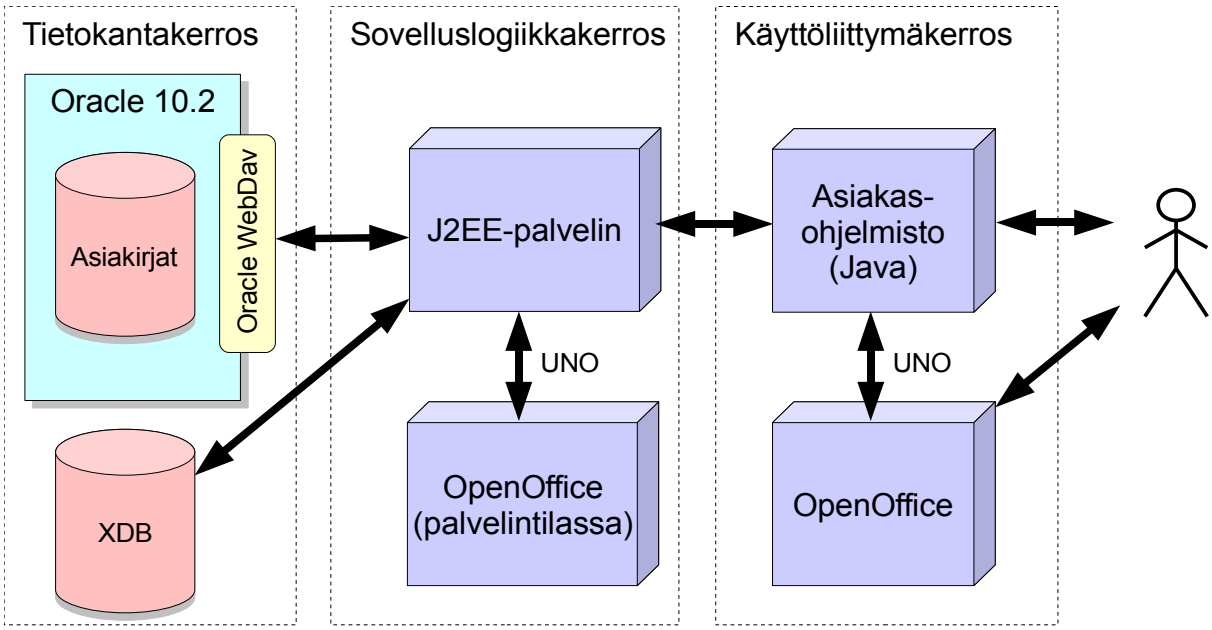


Kuva 1: Tietojärjestelmästä ulkoisella toimisto-ohjelmistolla käsiteltävät asiakirjat osana työnkulkua.

OpenDocument-tiedostomuodon käytöllä osana työnkulkua ohjaavaa tietojärjestelmää on merkitystä esimerkiksi silloin, kun järjestelmä luo asiakirjapohjan ja määrittelee pohjaan perustiedot, kuten yhteystiedot, diaarinumeron, muokkaushistorian ja muuta tietokantasisältöä. Tämän jälkeen käyttäjä kirjoittaa asiakirjan sisällön. Käsittelyn jälkeen asiakirja viedään tietojärjestelmän asiakirjanhallintaan tallennettavaksi. Järjestelmä voi poimia asiakirjasta metatietoja ja muuntaa sen automaattisesti toisiin asiakirjamuotoihin, tai liittää jotain metatietoja mukaan vasta tässä vaiheessa. Tämä prosessi on kuvattu kuvassa 1 yllä. Koko tallennuskin voidaan periaatteessa tehdä muussa kuin OpenDocument-muodossa. Integrointi toimisto-ohjelmistoon voi koskea muun muassa asiakirjan luomista tai avaamista ja sen toimittamista takaisin asiakirjanhallintajärjestelmälle tallennettaessa.

Esitämme esimerkkinä Slovenian oikeuslaitoksen asiakirjanhallintajärjestelmän, jossa

käytetään monikerrosarkkitehtuuriin pohjautuvaa ratkaisua. Arkkitehtuuri on kuvattu alla kuvassa 2. Arkkitehtuurin pohjana on Java EE -alustaa käyttävä sovelluspalvelin. Käyttäjän käyttöliittymänä käyttöliittymäkerroksessa toimii Java-pohjainen asiakasohjelma, sekä OpenOffice asiakirjojen muokkausta varten. Sovelluslogiikka on pääosin toteutettu Java EE -sovelluspalvelimella. Sovelluspalvelin käyttää OpenOfficea tiettyjä toimintoja, kuten PDF-muunnoksia varten. OpenOfficea käytetään palvelintilassa, joka tarkoittaa OpenOffice-prosessia, jolla ei ole käyttöliittymää ja jota voidaan komentaa UNO-rajapinnan kautta (ks. luku 11). Tietovarastokerroksessa käytetään Oracle- ja XDB-tietokantajärjestelmiä. Asiakirjasisältö tallennetaan Oracle-tietokantaan WebDav-asiakirjanhallintarajapintaa käyttäen.



Kuva 2: Slovenian oikeuslaitoksen asiakirjanhallinnan Java EE -sovelluspalvelimeen pohjautuva monikerrosarkkitehtuuri.

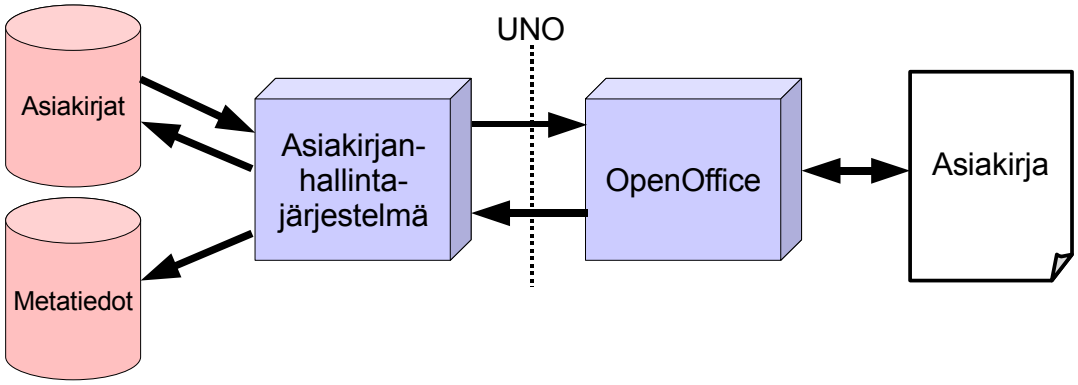
Toisena vastaavana esimerkkinä voidaan ottaa työryhmäohjelmisto Lotus Notes ja Lotus Domino -sovelluspalvelin. Näiden integrointi OpenOfficeen voidaan tehdä paljolti samalla tavoin kun edellä Slovenian Java EE -ratkaisussa. Vuonna 2007 ilmestyvän Lotus Notesin versio 8 sisältää OpenDocument-muotoa käsittelevän asiakasohjelman [IBM2007]. Lotus Notesille ja Dominolle on saatavilla myös kolmansien osapuolien tarjoamia OpenOffice-integrointiratkaisuja, kuten SWING Integrator.

Toimisto-ohjelmiston integroinnin tietojärjestelmän muihin osiin on oltava helppoa. Seuraavissa kahdessa luvussa esitetään kaksi toisiaan tukevaa lähestymistapaa toteuttaa integrointia: sovellusrajapinnat ja asiakirjamuodon muodostama rajapinta.

5.2. Integrointi sovellusrajapinnan avulla

Sovellusohjelmakohtaiset rajapinnat tarjoavat tehokkaan välineen hallita toimisto-ohjelmistoa. Esimerkiksi OpenOffice tarjoaa UNO-rajapinnan, jonka kautta lisäosa tai ulkoinen ohjelma voi hallita sitä. Esimerkiksi asiakirjanhallintajärjestelmä voi käynnistää OpenOfficen, komentaa sitä UNO-rajapinnan kautta luomaan uuden asiakirjan, esitäyttää asiakirjaan metatiedot ja avata sen käyttäjän muokattavaksi. Kun käyttäjä on muokannut asiakirjaa, voi asiakirjanhallintajärjestelmä lukea sen sisällön tai tallentaa sen sellaisenaan OpenDocument-

tai jossain muussa muodossa. Järjestelmä voi poimia käyttäjän muokkaamasta asiakirjasta metatietoja tai muuten indeksoida sen. Tietojärjestelmä voi käyttää sovellusrajapintoja myös asiakirjojen tulostamiseen ja muunnoksiin toisiin asiakirjamuotoihin.



Kuva 3: OpenOffice-integrointi UNO-rajapinnan avulla.

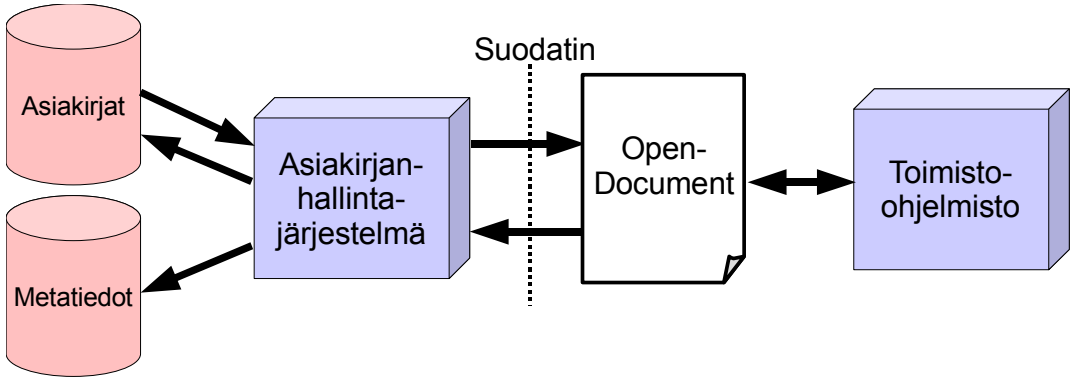
Tässä ratkaisussa asiakirjanhallintajärjestelmä ei välttämättä lainkaan käsittele OpenDocument-asiakirjan XML-muodossa olevaa sisältöä, vaan sovellusrajapinnan tarjoamaa näkymää asiakirjan sisällöstä, joka on tyypillisesti erilainen kuin OpenDocument-muotoon tallennettu rakenne. Tällöin on epäoleellista, onko asiakirja ylipäätään OpenDocument-muodossa vai jossain muussa.

UNO-sovellusrajapinnan käytöstä kerrotaan tarkemmin luvuissa 7.2 ja 11.

5.3. Asiakirjamuoto integroinnin ratkaisuna

Toinen ratkaisu on käyttää rajapintana asiakirjamuotoa. OpenDocument on standardoitu ja monien eri toimistosovellusten tukema asiakirjamuoto, toisin kuin esimerkiksi OpenOfficen sovelluskohtainen UNO-rajapinta. OpenDocument-asiakirjan käsittely voi useimmissa tapauksissa olla myös helpompaa kuin sovelluskohtaisten rajapintojen käyttö, kun halutaan käsitellä itse asiakirjasisältöä.

OpenDocument-asiakirjan vientiin ja tuontiin asiakirjanhallintajärjestelmästä käytetään suodinta, joka voi olla toteutettu esimerkiksi XSLT-muunnoskielellä tai muulla järjestelmän tukemalla kielellä. Asiakirjanhallintajärjestelmä luo asiakirjan tai asiakirjapohjan OpenDocument-muodossa, antaa sen käyttäjän muokattavaksi toimisto-ohjelmistossa ja tallennuksen jälkeen lukee asiakirjasta sen sisällön tai metatiedot takaisin tietojärjestelmään. Myös tässä ratkaisussa voidaan käyttää sovellusrajapintaa, mutta se rajoittuu vain asiakirjan avaamis- ja tallennuskomentojen tekemiseen, eikä sisällä asiakirjasisällön käsittelyä.



Kuva 4: Integrointi OpenDocument-asiakirjamuodon avulla.

Tietojärjestelmät voivat siis luoda asiakirjoja, asiakirjapohjia tai niiden osia tietokantasisällön pohjalta. Toisaalta ne voivat jäsentää ja poimia asiakirjojen sisältöä erilaisiin tarkoituksiin. OpenDocument-asiakirjat voivat myös muodostaa lomakkeita, jotka voivat olla suoraan yhteydessä tietokantaan tai lomakesisältö voidaan lukea asiakirjasisällöstä (ks. luku 5.9).

OpenDocument-muodon käytöstä ulkoisilla työkaluilla kerrotaan tarkemmin luvussa 10. Liitteessä B kerrotaan asiakirjojen käsittelystä XSLT-kielellä, jota voidaan käyttää myös toimisto-ohjelmistoista riippumattomasti.

5.4. Asiakirjamuunnokset

Asiakirjamuunnokset ovat OpenDocument-muodon tärkeä käyttötapaus, sillä eri tahot tarvitsevat eri tiedostomuodossa olevia asiakirjoja. Kaksi tavallisinta tapausta ovat muunnos PDF-muotoon ja muunnos Microsoft Office -tiedostomuotoihin ja takaisin; näitä muunnoksia varten ei juurikaan ole saatavilla toimisto-ohjelmista erillisiä työkaluja, joten ne tehdään yleensä käyttäen toimisto-ohjelmiston sovellusrajapintoja, tavallisimmin OpenOfficen UNO-rajapintaa. UNO-rajapintaa käyttämällä laajan asiakirjamuotovalikoiman tarjoavia muuntimia on esimerkiksi JODConverter, jota voidaan käyttää paitsi komentoriviltä, myös Java-palvelinsovelmana web-sovelluspalvelimessa. JODConverter käyttää OpenOfficea tällöin palvelintilassa (ks. luku 11). Lisätietoja JODConverter-ohjelmasta löytyy osoitteesta:

http://www.artofsolving.com/opensource/jodconverter

5.5. Metatietojen keruu

Metatietoja käytetään asiakirjojen indeksoinnissa. Metatietojen keruu liittyy usein luvussa 5.1 esitetyn kaltaiseen laajempaan käyttötapaukseen, jossa työnkulkua toteuttava tietojärjestelmä poimii asiakirjoista metatietoja, mutta sitä voi tehdä myös esimerkiksi työaseman paikallinen hakutoiminto.

OpenDocument sisältää joukon Dublin Core -standardin mukaisia metatietokenttiä, sekä joukon omia kenttiä (ks. luku 6.5). Osa näistä kentistä on sellaisia, jotka asiakirjan kirjoittajan täytyy itse syöttää. Hankaluutena on, että ohjelmistojen tarjoama käyttöliittymä niiden syöttämiseksi on vaikeahkosti käytettävä. Esimerkiksi OpenOfficessa metatietokenttien sisältöä pääsee muokkaamaan vain valikon kautta. Kuitenkin käyttäjän kannalta olisi parempi, jos kenttien sisällön voisi kirjoittaa suoraan niille varattuun kohtaan. Tämän vuoksi metatietokentät toteutetaan usein jollain käyttäjälle helpommalla tavalla. Mahdollisia merkintätapoja metatiedoille ovat ainakin syöttökentät, lomakkeen ohjausobjektit, taulukon

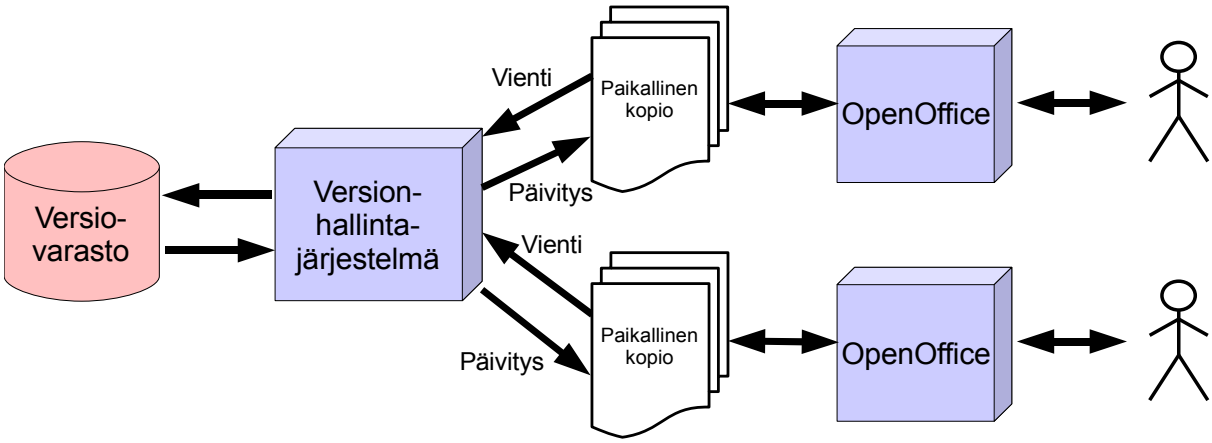
solut ja muuten merkittyjen syöttökohtien käyttö. Metatiedot saatetaan merkitä esimerkiksi tietyllä kappaletyylillä; tämä on toimiva tapa ainakin otsikoiden ja tiivistelmän suhteen.

Yllä esitetyn kaltainen metatietojen epästandardi toteutustapa voi muodostaa ongelmia, sillä indeksointia suorittavien järjestelmien täytyy tuntea käytetty merkintätapa. Yleisissä indeksoijissa ei voida ottaa huomioon kaikkia mahdollisia metatietojen esitystapoja, vaan joudutaan tyytymään standardin mukaisiin. Jos metatiedot on tallennettu epästandardilla käytännöllä, ei niitä löydy. Siksi eräs metatietojen käsittelyn tehtävä voi olla asiakirjoissa epästandardilla tavalla merkittyjen metatietojen muuntaminen tai kopioiminen standardin mukaisiksi metatietokentiksi. Tätä ongelmaa ei välttämättä ole hakukoneiden indeksoijissa, jos ne käyttävät vain XML:n tekstisisältöä kontekstista välittämättä. Esimerkiksi Google Desktop Search -hakuohjelmiston OpenDocument-jäsennyskomponentti ei tällä hetkellä huomioi tekstin kontekstia hakutulosten painotuksessa, vaan käsittelee metatietoelementtien sisällön tavallisena tekstinä.

5.6. Versionhallinta

Versionhallinta on usein käytetty asiakirjahallinnan tekniikka erityisesti ryhmätyöskentelyssä. Asiakirjan eri versiot voidaan tallentaa versionhallintajärjestelmään, jolloin vanhat versiot ovat helposti palautettavissa. Versioihin voidaan myös tallentaa versiotietoja, kuten muokkaaja, kommentteja ja muita metatietoja.

Monet ryhmätyö- ja asiakirjanhallintajärjestelmät, kuten Lotus Notes, tarjoavat mahdollisuuksia asiakirjojen tallentamiseen jonkinlaisella versiointituella. Lisäksi on varsinaisia versionhallintajärjestelmiä, kuten Subversion, CVS, BitKeeper ja ClearCase. Nämä järjestelmät ovat yleensä erikoistuneet muotoilemattomien tekstitiedostojen tallentamiseen, versiointiin ja samanaikaiseen muokkaukseen eri käyttäjien toimesta. Ne soveltuvat myös binääritiedostojen (kuten pakatussa muodossa olevien OpenDocument-asiakirjojen) tallennukseen, mutta binääritiedostojen samanaikainen muokkaus ei yleensä ole käytännössä mahdollista.

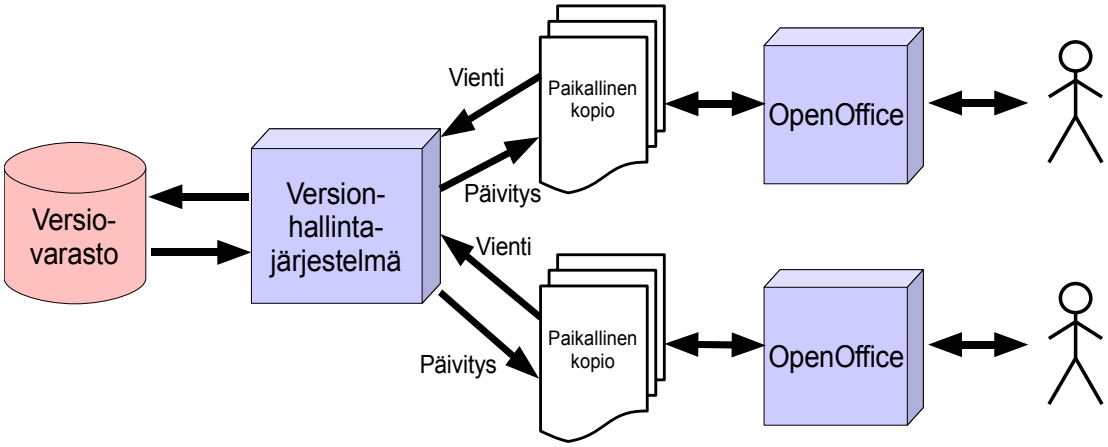


Kuva 5: Versionhallinta toimii usein siten, että versiovaraston uusimmista asiakirjaversioista otetaan paikallinen kopio kunkin käyttäjän tietokoneelle. Jos käyttäjä tekee asiakirjoihin muutoksia, täytyy muutokset viedä versionhallintaan. Muut käyttäjät saavat muutokset käyttöönsä päivittämällä oman kopionsa.

OpenDocument tukee itsessään tiettyjä versionhallintatoimintoja, kuten muutoshistorian keräämistä ja asiakirjan eri versioiden vertailua siten, että toimisto-ohjelmisto voi korostaa tehdyt muutokset. Myös eri versioiden yhdistäminen on mahdollista. Näitä toiminnallisuuksia

voidaan käyttää yhdessä versionhallintajärjestelmän kanssa manuaalisesti, mutta parempi ratkaisu on integrointi versionhallintajärjestelmään. OpenDocument-asiakirjojen vertailulle ja yhdistämiselle on tuki ainakin TortoiseSVN-ohjelmalle Windows-ympäristössä; ohjelma käyttää OpenOfficen UNO-rajapintaa.

Ryhmätyökäytössä on kaksi tekniikkaa asiakirjojen muokkaukselle. Jos useat käyttäjät haluavat muokata samaa asiakirjaa, se voidaan ensinnäkin lukita muokattavaksi yhdelle käyttäjälle kerrallaan. Toinen, kehittyneempi tapa on, että eri käyttäjät voivat muokata asiakirjaa samanaikaisesti. Kun asiakirjat viedään versionhallintajärjestelmään, niihin tehdyt muutokset yhdistetään (merge). Yhdistämisprosessi on esitetty kuvassa 6 alla. Yhdistäminen toimii kuitenkin vain tiettyjen asiakirjamuotojen kanssa ja OpenDocumentille ei toistaiseksi ole kunnollista tukea. Käyttäjän täytyy tehdä vertailu ja yhdistäminen itse manuaalisesti, mikä on jossain määrin vaivalloista.



Kuva 6: Muokattujen asiakirjaversioiden vertailu vanhaan asiakirjaan ja yhdistäminen. Vertailua vanhaan asiakirjaan ei tarvitse tehdä, jos muokattaessa on käytetty muutoshistorian tallennusta.

5.7. XML-tietokantaraportit

Ulkoisesta tietolähteestä, kuten tietokannasta, halutaan usein tuoda sisältöä asiakirjoihin tai luoda kokonaisia asiakirjoja. OpenDocument-muotoisen tietokantaraportin luominen saattaa olla tarkoitus itsessään tai toimia vain integrointiratkaisun välimuotona tulostusoperaatiossa tai muunnoksessa esimerkiksi PDF-muotoon. Tietokantaraporttien tekemiseen on muutamia perusratkaisuja riippuen jälleen siitä, käytetäänkö toteutukseen ohjelmistojen sovelluskohtaisia rajapintoja vai tiedostomuotoa.

Yksinkertaiset tietokantaraportit voidaan usein toteuttaa toimisto-ohjelmistojen tietokantaominaisuuksilla. OpenDocument määrittelee tietokantakenttiä, jotka mahdollistavat asiakirjan käytön raporttipohjana. Esimerkiksi OpenOffice Base -tietokantaraportit ja Writer-joukkokirjeet toimivat tällä periaatteella. Tätä ratkaisua rajoittavat lähinnä ohjelmiston raportti- ja tietokantayhteysominaisuudet.

Raporttien luomiseen vaikuttaa ratkaisevasti tietokantojen tyyppi. Tavanomaiset relaatiotietokannat eivät ylipäätään ole kovin soveltuvia esittämään luonteeltaan vahvasti rakenteista tietoa. Tiedon tallentaminen XML-muodossa, kuten XML-tietokantaan, on eräs ratkaisu tähän. Toisaalta ehkä tavallisempaa on käyttää XML-muotoa vain välimuotona. Koska sitä jo muutenkin käytetään paljon tiedon vaihtoon tietojärjestelmien välillä, on vienti tietokannasta XML-muotoon usein jo valmiina.

OpenDocument tarjoaa mahdollisuuden tehdä hyvinkin monipuolisia tietokantaraportteja muuntamalla XML-tiedostoja OpenDocument-muotoon esimerkiksi XSLT-suotimella. Koska XSLT-suotimet on mahdollista asentaa OpenOfficeen tuonti- ja vientisuotimiksi ja käyttäjä voi yksinkertaisesti avata tietyn muotoisen XML-tiedoston suoraan, tarjoaa se mahdollisuuden rakentaa myös käyttäjän kannalta helppoja ratkaisuja.

Luvussa 9 esitetään esimerkki XML-muodossa olevan henkilötietokannan muunnoksesta OpenDocument-asiakirjaksi tulostusta ja PDF-vientiä varten.

5.8. Asiakirjojen sisällön muuttaminen

Jos asiakirjassa on suuri määrä tiettyjä kohtia, jotka halutaan muuttaa toisenlaiseksi, tai jos asiakirjoja on suuri määrä, ei muutoksia kannata tehdä käsityönä. Toimisto-ohjelmistot tarjoavat jonkin verran mahdollisuuksia yhdessä asiakirjassa olevien kohtien muuttamiseen, kuten etsi ja korvaa -toiminnon. Joskus vaadittavat muutokset ovat kuitenkin vaikeita tehdä vain korvauksella tai asiakirjoja saattaa olla paljon. Tyypillinen esimerkki on linkitettyjen kuvien kansiopolun muuttaminen. Tällaisiin tapauksiin ovat ratkaisuna makrot, ulkoiset sovellusrajapintaa, kuten UNO-rajapintaa, käyttävät ohjelmat ja OpenDocument-asiakirjojen muuntaminen erillisen ohjelman avulla.

ZIP-pakettina olevasta OpenDocument-asiakirjasta on helppo avata XML-tiedostot ja muokata niiden sisältöä. Muokkaus voidaan tehdä XSLT-muunnoksilla, XML-ohjelmointirajapintojen avulla tai yleisillä tekstimuotoisten tiedostojen käsittelytyökaluilla, kuten Perl-, Python ja Awk-kielillä tai sed-ohjelmalla. Käsittelyn jälkeen sisältö voidaan jälleen kirjoittaa OpenDocument-tiedoston ZIP-pakettiin. Asiakirjan sisällön muokkaamisesta eri ohjelmointikielillä ja muilla työkaluilla kerrotaan luvussa 10.

5.9. Sähköiset lomakkeet

Sähköisillä lomakkeilla on monia käyttötapauksia, jotka asettavat erilaisia tarpeita lomakkeiden toteutusalustalle. Web-pohjaiset lomakkeet ovat monessa tapauksessa käyttäjän kannalta kenties helpoin tapa täyttää lomakkeita, koska WWW on pitkälti standardoitu ja mikäli palvelut on tehty näitä standardeja noudattaen, löytyy yhteensopiva selainohjelmisto käytännössä jokaisesta tietokoneesta. Täytettyjä web-pohjaisia lomakkeita ei kuitenkaan voi tallentaa paikallisesti, mikä olisi usein tarpeellista arkistoinnin kannalta. Niitä ei tyypillisesti voi tallentaa ja avata uudelleen edes niitä tarjoavassa järjestelmässä, vaan täyttäminen täytyy tehdä yhdellä kerralla. Web-lomakkeita ei voi myöskään tulostaa siten, että lomake sivuttuisi A4-arkille oikein, ellei tulostusmahdollisuutta ole erikseen lisätty. Näistä syistä johtuen käytetään paljon ladattavia lomakeasiakirjoja asiakirjamuodossa, joka mahdollistaa lomakkeen tarkan asettelun.

Alla on lueteltu tavallisimmat ladattavien sähköisten lomakkeiden käyttötavat edeten manuaalisimmasta automatisoiduimpaan, sen jälkeen kun käyttäjä on ladannut ja täyttänyt lomakkeen:

1. Käyttäjä tulostaa lomakkeen ja palauttaa tulosteen. Käsittelijä vie tiedot järjestelmään manuaalisesti.
2. Käyttäjä palauttaa täytetyn tiedoston, josta käsittelijä vie tiedot järjestelmään manuaalisesti.

3. Käyttäjä palauttaa täytetyn tiedoston, josta tiedot viedään järjestelmään lukemalla ne suoraan asiakirjatiedostosta automatisoidusti.
4. Täytetty lomake lähettää itse tietonsa järjestelmään.

OpenDocument-asiakirjat sopivat kaikkiin näihin käyttötapauksiin. Kaikissa niissä voidaan tehdä lomakkeiden esitäyttö järjestelmässä, josta lomake saadaan. Kohdan 3 tapauksen toteuttamista käsitellään luvussa 8, jossa näytetään, kuinka vientisuotimella voidaan poimia lomakekenttien tiedot OpenDocument-asiakirjasta. Kohdan 4 tapaus voidaan toteuttaa OpenDocument-asiakirjoilla useammallakin eri tavalla: tietokantaan kytketyllä lomakkeella, sisältönsä web-kyselynä lähettävällä tavanomaisella lomakkeella tai XForms-lomakkeella.

OpenDocument-lomakkeiden toteuttamisesta XForms-lomakkeina kerrotaan tarkemmin luvussa 12. Luvussa esitetyt XForms-lomakkeiden edut ja haitat suhteessa muihin ratkaisuihin pätevät paljolti myös ei-älykkäisiin OpenDocument-lomakkeisiin.

5.10. Tekstikatkelmien kokoaminen

Asiakirja voidaan kirjoittaa toisen asiakirjan sisälle katkelmina, jotka myöhemmin kootaan automaattisesti yhdeksi asiakirjaksi. Tällöin sisältävä asiakirja sisältää yleensä kommentteja sisälletystä asiakirjasta. Tätä tekniikkaa voidaan käyttää esimerkiksi tekstin valmisteluun kommentteja sisältävän työasiakirjan sisällä. Kokoaminen voidaan tehdä makrolla, vientisuotimella tai ulkoisella muuntimella.

OpenDocument-tiedostomuodon määrittelevän OASIS-standardin asiakirja muodostaa itsessään esimerkin tästä käyttötapauksesta. Standardi on kirjoitettu OpenDocument-asiakirjana, joka sisältää standardin Relax NG -rakennemäärittelyn (ks. liite E) seuraavanlaisina katkelmina:

```
220 <define name="office-drawing-content-prelude">
221     <ref name="text-decls"/>
222     <ref name="table-decls"/>
223 </define>
```

Määritykset jatkuvat kuvailutekstin jälkeen seuraavasti:

```
224 <define name="office-drawing-content-main">
225     <zeroOrMore>
226         <ref name="draw-page"/>
227     </zeroOrMore>
228 </define>
```

Käsittelyohjelma kokoaa tällaiset muun tekstin joukossa olevat, erityisellä kappaletyylillä merkityt pätkät ja kirjoittaa ne yhdeksi tiedostoksi. Tämän tyyppinen katkelmien kokoaminen on helppoa suorittaa XSLT-muuntimella ainakin, jos kohdetiedostomuotona on muotoilematon teksti.

5.11. Tekstin automatisoitu korostus

Monissa teknisissä teoksissa on jollain muodollisella kielellä, kuten ohjelmointikielellä, kirjoitettuja katkelmia. Tässäkin raportissa on lukuisia XML-muotoisia ja jollain ohjelmointikielellä kirjoitettuja listauksia. Raportin XML-listauksissa on elementtien attribuuttien arvot, elementtien sisältämä teksti ja XML-kommentit kirjoitettu käyttäen tiettyjä

kappale- ja merkkityylejä, joilla on tietty värimääritys. Tämän tyyppiset tehtävät, joita kutsutaan kieliopin korostukseksi, on melko helppoa automatisoida asiakirjamuunnoksina.

6. OPENDOCUMENT TIEDOSTOMUOTONA

OASIS OpenDocument Format pohjautuu World Wide Web Consortium -yhtymän kehittämään XML-merkintäkieleen. XML-kieli sopii parhaiten rakenteellisen tiedon esittämiseen ja helpottaa asiakirjoja käsittelevien ohjelmien ylläpitoa ja kehittämistä, sillä sen käsittelyyn on olemassa valtavasti työkaluja. Lisäksi XML-tiedostot ovat tekstitiedostoina edes periaatteessa ihmisen luettavia, toisin kuin binääritiedostot, ja niitä voi tarpeen tullen muokata tekstieditorilla. Standardoidun XML-muodon ansiosta asiakirjojen rakenne voidaan pitää erillään ohjelmien sisäisestä esitysmuodosta. XML-merkintäkielestä annetaan lyhyt johdanto liitteessä A.

David Eisenbergin verkkokirja "OASIS OpenDocument Essentials" [Eis2006] antaa hyvän yleisesityksen OpenDocument-tiedostomuodosta ja sen automaattisesta käsittelystä. OpenDocument-tiedostomuodon yleisrakenteesta on lisäksi Wikipedia-sivu seuraavassa verkko-osoitteessa:

http://en.wikipedia.org/wiki/OpenDocument_technical_specification

6.1. Yleispiirteitä

OpenDocument-tiedostomuodolla on XML-kielenä tiettyjä piirteitä, joista alla on lueteltu joitain keskeisimpiä. Näillä piirteillä on merkitystä toisiin muotoihin muunnettavuuden kannalta, sekä yleensäkin yhteensopivuuden kannalta muiden standardien kanssa. Vertaamme tiedostomuotoa muun muassa web-sivustoissa käytettäviin HTML- ja XHTML-merkintäkieliin ja teknisessä dokumentoinnissa käytettävään DocBook-merkintäkieleen.

Helpoiten havaittava piirre OpenDocument-asiakirjoissa on, että ne ovat ZIP-pakattuja tiedostoja, jotka sisältävät joukon XML-tiedostoja. Itse XML-esityskin on siten pilkottu erillisiin tiedostoihin. Esimerkiksi tyylien eriyttämisestä erilliseen tiedostoon on etua tuontisuodinten rakentamisessa. Lisäksi esimerkiksi asiakirjoihin liitetyt kuvat ovat ZIP-paketissa binäärimuodossa erillisinä tiedostoina, mikä on käsittelyn helppouden ja tilankäytön kannalta hyvä ratkaisu. OpenDocument-asiakirja voidaan kuitenkin esittää myös yhden XML-tiedoston muodossa (ks. luku 6.8), jota tapaa käytetään vienti- ja tuontisuodattimissa.

OpenDocument on XML-rakenteeltaan paljolti *sekasisältöinen*, toisin sanoen tekstin välissä voi olla elementtejä. Esimerkiksi kappaleen teksti on `<text:p>`-elementin sisällä ja sen seassa voi olla muita elementtejä seuraavasti:

```
<text:p text:style-name="Standard">Tässä on tekstiä, jonka <text:span text:style-
name="Strong_20_Emphasis"> välissä on lihavointia</text:span> ja manuaalisen
rivinvaihdon elementti<text:line-break/> ja taas tekstiä</text:p>
```

Tämä tekee OpenDocument-muodosta helposti muunnettavan muihin sekasisältöisiin tiedostomuotoihin, kuten HTML, XHTML tai DocBook. Sekasisältöisyys on erityisen tärkeää joissain käyttökohteissa, kuten asiakirjojen kääntämisessä työkaluilla, jotka poimivat asiakirjoista XML-elementtejä. Esimerkiksi kappale muodostaa luonnollisen yksikön käännettäväksi. Jos kappaleen sisällä on muita kuin tekstielementtejä, ne voidaan korvata tekstin joukossa tyyliin: ”`Kiinan kielen merkki <element id="42"/> kuvaa lempeyttä.`” Tiedostomuotoa, joka ei ole sekasisältöinen, on vaikea käyttää tämän tyyppisissä sovelluksissa. Niissä teksti jakautuu luonnottomiin katkelmiin, kuten ”`Kiinan kielen merkki`”, jota seuraa välissä oleva elementti ja ” `kuvaa lempeyttä.`” Nämä

katkelmat täytyisi kääntää erikseen, ei kokonaisena virkkeenä, mikä vaikeuttaisi kääntämistä merkittävästi.

OpenDocument erottaa sisältöä ja esitystä siten, että kaikki muotoiluasetukset tehdään tyylejä käyttäen. OpenDocument-muodossa ei ole HTML-tyyppisiä tyylikohtasia elementtejä, kuten `<b>`, jolla merkitään elementin sisällä olevan tekstin olevan lihavoitua. Tekstiasiakirjoissa tyylit asetetaan `<text:p>`- ja `<text:span>`-elementeissä. Mikäli tekstiin tehdään kohdekohtaisia muotoiluja varsinaisia tyylejä käyttämättä, kuten kursivointia tai lihavointia, luodaan *automaattinen tyyli*, jossa kohdekohtaiset asetukset määritellään. Tämän ansiosta OpenDocumentissa ei tarvitse olla erityisiä elementtejä esimerkiksi kursivoidulle tai lihavoidulle korostukselle. Tämä on tärkeää, koska korostukset ovat fonttien ominaisuuksia. Fontit sisältävät usein erikseen kursiivin ja konekursiivin tai useita eri lihavointivahvuuksia. Tapa myös vähentää OpenDocument-standardin XML-elementtien määrää. Lisäksi samanaikaiset korostukset, kuten samanaikainen lihavointi ja kursiivi, saadaan yhdistettyä saman automaattisen tyylin sisälle.

OpenDocument on tyylien ja useimpien muiden piirteiden suhteen *hierarkkinen,* mikä tarkoittaa, että esimerkiksi tekstin tyyli saadaan selvitettyä sen sisältävistä hierarkian ylemmän tason elementeistä ilman, että koko asiakirjaa täytyy käydä lävitse alusta kyseiseen kohtaan. Esimerkiksi lihavointi, joka jatkuu kappaleesta toiseen, suljetaan ensimmäisen kappaleen lopussa ja avataan taas seuraavassa kappaleessa. HTML-kielessä, joka on myös hierarkkinen, tämä ilmaistaisiin seuraavasti:

```
<p>Lihavoimatonta tekstiä, <b>joka muuttuu lihavoiduksi.</b></p>
<p><b>Seuraavassa kappaleessa lihavointi aloitetaan taas</b> uudestaan.</p>.
```

Huomattakoon, että mikäli lihavointi jatkuu tekstiasiakirjassa yli kokonaisten kappaleiden, tehdään lihavointi kappaleiden muotoiluasetuksiin. Sitä ei siten tehdä väliin jäävissä kappaleissa kappaleen alusta kappaleen loppuun. Muunnokset hierarkkisten kielten välillä ovat hyvin helppoja verrattuna muunnoksiin hierarkkisten ja epähierarkkisten kielten välillä. Hierarkkisuus on XML:n kieliopillinen peruspiirre (ks. liite A), mutta XML-asiakirjoissa on mahdollista olla merkintöjä, joiden semanttinen merkitys ei ole hierarkkinen. Erityisesti OpenDocument-muodossa ei ole erikoismerkintöjä, joiden avulla kierrettäisiin XML:n perusominaisuutta, että elementit ovat aina sisäkkäisiä, eivät koskaan lomittaisia. Lomittaisia merkintöjä sisältävästä XML-muodosta olisi käytännössä mahdotonta tehdä muunnoksia esimerkiksi XSLT-muunnoskielen avulla.

Tietyt ominaisuudet, kuten numerointi, määritelmänsä mukaan vaativat tietynlaista muistillisuutta, jossa täytyy ylläpitää asiakirjan alusta alkavaa laskuria. Tämänkaltainen muistillisuuskin rikkoo hierarkkisuutta. Harvat muistilliset ominaisuudet, kuten numerointi, ovat kuitenkin yleensä tuettuja kaikissa saman aihealueen merkintäkielissä. Eräs vastaava epähierarkkinen piirre on jäsennystaso; tekstin jäsennystason saa selville vain etsimällä edeltävän `<text:h>`-elementin, jonka `text:outline-level`-attribuutti kertoo sisennystason. Hierarkkinen ratkaisu löytyy esimerkiksi DocBook-kielestä, jossa jäsennystaso-elementit kirjoitetaan sisäkkäisesti seuraavanlaisesti: `<chapter> <section> <section> </section> </section> </chapter>`. Jäsennystason epähierarkkisuudella ei ole juurikaan merkitystä, sillä jäsennysnumerointi on joka tapauksessa väistämättä epähierarkkista. Myös XHTML-kieli on otsikoiden jäsennystason tapauksessa epähierarkkinen.

OpenDocument-muodon XML-merkinnät koskevat pääasiassa muotoilua, eikä niillä yleensä merkitä tekstin asiayhteyttä. OpenDocument-muodossa ei esimerkiksi ole elementtejä

erityisesti henkilöiden nimiä tai osoitteita varten, kuten DocBook-kielessä on. Tällaiset asiayhteyden mukaiset merkinnät tehdään yleensä tyyleillä. Joitain asiayhteyskohtaisia elementtejä kuitenkin on, muun muassa metatietoelementit ja jäsennysotsikot.

OpenDocument-tiedostomuoto käyttää *nimiavaruuksia* vahvasti ja systemaattisesti. Nimiavaruudet ovat XML:n piirre, jolla estetään saman nimiset elementit silloin, kun samassa XML-tiedostossa käytetään usean eri XML-muodon elementtejä (ks. Liite A). OpenDocument-tiedostomuodossa tämä mahdollistaa olemassa olevien XML-pohjaisten standardien helpon käytön ilman ristiriitaongelmia (ks. luku 6.2 alla). Nimiavaruudet käytännössä pidentävät elementtejä hieman, mutta tällä ei ole merkitystä silloin, kun tiedostomuotoa käsitellään pääsääntöisesti ohjelmallisesti. Esimerkiksi XHTML- ja DocBook-tiedostomuodoissa nimiavaruuksia ei yleensä käytetä, koska niitä ei tyypillisesti kirjoiteta ohjelmallisesti vaan ihmisen toimesta.

6.2. Perustuminen standardeihin

OpenDocument ISO-standardi pohjautuu vahvasti olemassa oleviin muihin avoimiin kansainvälisiin standardeihin. Esimerkiksi erilaisissa tiedon esityksissä käytetään lukuisia muita ISO-standardeja, kuten kielten koodit (ISO 639), maakoodit (ISO 3166) ja aikojen ja päivämäärien esitys (ISO 8601).

Asiakirjojen metatiedot esitetään Dublin Core -metatietostandardin (ANSI/NISO Z39.85-2001) alijoukkona. Dublin Core -standardia ovat Suomesta olleet kehittämässä Kansalliskirjasto ja Helsingin yliopiston kirjasto. Tämän standardin käyttö on erityisen tärkeää, koska mikä tahansa Dublin Core -standardia ymmärtävä ohjelmisto pystyy helposti poimimaan OpenDocument-asiakirjoista metatiedot avattuaan niiden ZIP-pakatun tiedoston. Dublin Core -metatietojen käytöstä OpenDocument-standardissa kerrotaan tarkemmin luvussa 6.5.

OpenDocument käyttää monia web-standardeja määrittelevän W3C-konsortion julkaisemia standardeja. Matemaattiset kaavat esitetään MathML-merkintäkielellä. Viittaukset ulkoisiin kohteisiin, kuten grafiikkatiedostoihin, tehdään XLink-viittauskielellä. Animoiduissa esityksissä käytetään SMIL-multimediakieltä (Synchronized Multimedia Integration Language), joka määrittää esitysten ajoitus-, siirtymä- ja muita ominaisuuksia. XSL FO -muotoilukieltä (Formatting Objects) on lainattu sivu- ja tekstinasettelun merkitsemiseen. Grafiikkaobjektien attribuuteissa käytetään vahvasti vektorigrafiikan kuvauskieltä SVG:tä (Scalable Vector Graphics). OpenDocument käyttää lisäksi älykkäiden lomakkeiden toteuttamiseen tarkoitettua XForms-kieltä ja sen käyttämää XPath-kyselykieltä (ks. luku 12). Monien näiden standardien käyttö on osittaista: esimerkiksi SVG:stä käytetään vain attribuutteja, ei varsinaisia piirroselementtejä. Näiden eri XML-pohjaisten kielten elementeissä käytetään kyseisten standardien mukaisia nimiavaruuksia.

6.3. OpenDocument-tiedoston yleisrakenne

OpenDocument-standardi määrittelee kaikkiaan 16 eri asiakirjatyyppiä, joilla kaikilla on oma tiedoston nimen yhteydessä käytettävä kolmen kirjaimen tarkenne. Seuraavassa luettelossa ovat standardin mukaiset tiedostomuodot, niiden tiedostotarkenteet ja tiedostojen Internet-standardia noudattavat MIME-tyypit (Multipurpose Internet Mail Extensions).

Asiakirjatyyppi	Tiedosto-tarkenne	MIME-tyyppi
Tekstiasiakirja	.odt	application/vnd.oasis.opendocument.text
Tekstiasiakirjan malli	.ott	application/vnd.oasis.opendocument.text-template
Laskentataulukko	.ods	application/vnd.oasis.opendocument.spreadsheet
Laskentataulukon malli	.ots	application/vnd.oasis.opendocument.spreadsheet-template
Esitys	.odp	application/vnd.oasis.opendocument.presentation
Esityksen malli	.otp	application/vnd.oasis.opendocument.presentation-template
Piirros	.odg	application/vnd.oasis.opendocument.drawing
Piirroksen malli	.otg	application/vnd.oasis.opendocument.drawing-template
Kaavio	.odc	application/vnd.oasis.opendocument.chart
Kaavion malli	.otc	application/vnd.oasis.opendocument.chart-template
Kaava	.odf	application/vnd.oasis.opendocument.formula
Kaavan malli	.otf	application/vnd.oasis.opendocument.formula-template
Kuva	.odi	application/vnd.oasis.opendocument.image
Kuvan malli	.oti	application/vnd.oasis.opendocument.image-template
Verkkosivun malli	.oth	application/vnd.oasis.opendocument.text-web
Perusasiakirja	.odm	application/vnd.oasis.opendocument.text-master

Kaikki nämä asiakirjatyypit kuitenkin noudattavat samaa perusrakennetta. OpenDocument-asiakirja on ZIP-pakkauksella tiivistetty tiedosto, joka sisältää kokoelman tekstimuotoisia XML-tiedostoja. Lisäksi paketti voi sisältää esimerkiksi asiakirjaan lisättyjä kuvatiedostoja, jotka tallennetaan binäärimuotoisina tiedostoina, tyypillisesti JPEG- tai PNG-muodossa tiivistettyinä. Asiakirjapaketin tärkeimmät tiedostot ja kansiot ovat:

Tiedosto	XML-juurielementti	Kuvaus
`mimetype`	-	Asiakirjan tiedostomuodon MIME-tyyppi (teksti, taulukko, jne)
`content.xml`	`office:document-content`	Asiakirjan varsinainen sisältö
`styles.xml`	`office:document-styles`	Asiakirjan tyylit (luettelo-, kappale-, merkkityylit, jne.)
`meta.xml`	`office:document-meta`	Asiakirjan metatiedot (tekijä, pvm, viimeinen muokkaaja, jne.)
`settings.xml`	`office:document-settings`	Asiakirjakohtaiset asetukset (ikkunakoko, kohdistimen paikka, jne.)
`Pictures/`	-	Asiakirjan kuvat sisältävä kansio
`META-INF/manifest.xml`	`manifest:manifest`	Sisällysluettelo ZIP-paketin tiedostoista

Asiakirjatyyppi – mimetype

Tiedosto `mimetype` on tekstimuotoinen tiedosto, joka määrittelee asiakirjan tyypin MIME-tyyppitunnisteella. Asiakirjatyyppien MIME-tunnisteet on annettu edellä sivun 21 taulukossa.

6.4. Asetukset – settings.xml

Tiedosto `settings.xml` määrittelee asiakirjan sovelluskohtaiset asetukset. Näillä asetuksilla on merkitystä vain asiakirjan luoneelle toimistosovellukselle, kuten OpenOfficelle.

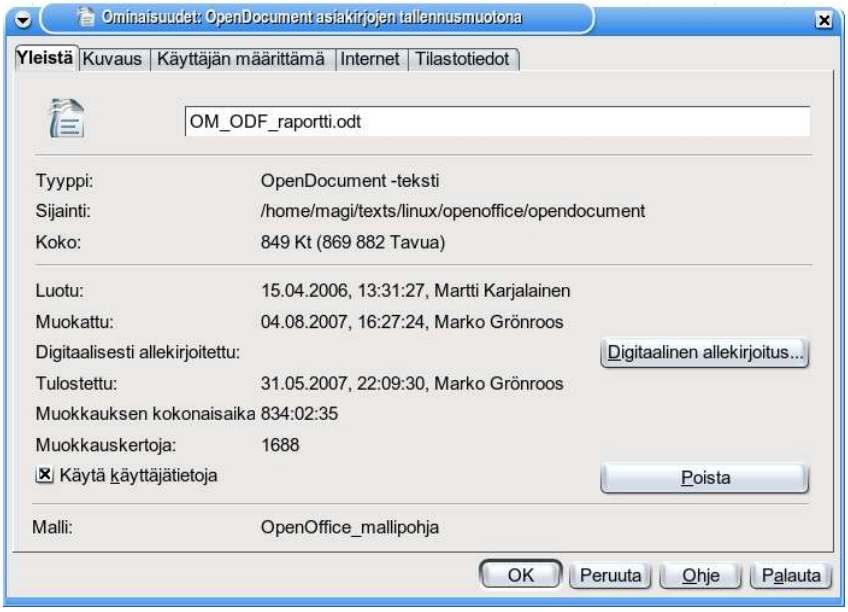
Tiedoston ylimpänä elementtinä on `<office:document-settings>`, joka sisältää nimiavaruusmäärittelyt, ja jonka alla on vain yksi elementti, `<office:settings>`. Sen alla on `<config:config-item-set>`-asetusjoukkoja, jotka sisältävät `<config:config-item>`-asetustietueita.

```
<office:document-settings
   xmlns:office="urn:oasis:names:tc:opendocument:xmlns:office:1.0"
   xmlns:xlink="http://www.w3.org/1999/xlink"
   xmlns:config="urn:oasis:names:tc:opendocument:xmlns:config:1.0"
   xmlns:ooo="http://openoffice.org/2004/office"
   office:version="1.0">
   <office:settings>
      <config:config-item-set config:name="ooo:configuration-settings">
         <config:config-item config:name="PrinterName"
            config:type="string">Generic Printer</config:config-item>
         ... lisää asetustietueita ...
      </config:config-item-set>
      ... lisää asetusjoukkoja ...
   </office:settings>
</office:document-settings>
```

Asetustietueen nimi määritellään `config:name`-attribuutilla ja tyyppi `config:type`-attribuutilla. Itse asetuksen arvo annetaan elementin sisällä.

6.5. Metatiedot – meta.xml

Tiedosto `meta.xml` sisältää asiakirjan metatietoja, joista useimmat löytyvät OpenOfficessa valikosta **Tiedosto > Ominaisuudet**.



Kuva 7: Asiakirjan metatietoja.

Tiedoston juurielementtinä on `<office:document-meta>`, joka sisältää

nimiavaruusmäärittelyt, ja jonka alla on vain yksi elementti, `<office:meta>`. Sen alla on `meta`- ja `dc`-nimiavaruuden elementtejä, joissa määritellään metatietoja.

```
<office:document-meta
   xmlns:office="urn:oasis:names:tc:opendocument:xmlns:office:1.0"
   xmlns:xlink="http://www.w3.org/1999/xlink"
   xmlns:dc="http://purl.org/dc/elements/1.1/"
   xmlns:meta="urn:oasis:names:tc:opendocument:xmlns:meta:1.0"
   xmlns:ooo="http://openoffice.org/2004/office" office:version="1.0">
   <office:meta>
      <meta:generator>OpenOffice.org/2.2$Linux
         OpenOffice.org_project/680m14$Build-9134
      </meta:generator>
      <meta:initial-creator>Martti Karjalainen</meta:initial-creator>
      <meta:creation-date>2006-04-15T13:31:27</meta:creation-date>
      <dc:creator>Marko Grönroos</dc:creator>
      <dc:date>2007-05-14T13:44:58</dc:date>
      ... muita metatietoja ...
   </office:meta>
</office:document-meta>
```

Metatietojen `dc`-elementit ovat Dublin Core -metatietoja, jotka pohjautuvat ISO 15836:2003 -standardiin. Tärkeimmät Dublin Core -metatietoelementit on luetteloitu alla. Luettelossa on annettu sulkeissa elementtiä vastaava välilehti ja kohta **Tiedosto > Ominaisuuksia** -ikkunassa.

`<dc:title>`	Asiakirjan otsikko (**Kuvaus > Otsikko**)
`<dc:subject>`	Asiakirjan aihe (**Kuvaus > Aihe**); Dublin Core -standardissa avainsanat
`<dc:description>`	Kommentteja (**Kuvaus > Huomautuksia**)
`<dc:creator>`	Asiakirjan viimeisin muokkaaja (**Yleistä > Muokattu**)
`<dc:date>`	Viimeisimmän muokkauksen päivämäärä ja -aika (**Yleistä > Muokattu**)

OpenDocument-standardissa on lisäksi lukuisia `meta`-nimiavaruudessa määriteltyjä muita metatietoelementtejä. Luettelossa alla on annettu sulkeissa elementtiä vastaava välilehti ja kohta **Tiedosto > Ominaisuuksia** -ikkunassa.

`<meta:generator>`	Asiakirjan luoneen ohjelman tunniste.
`<meta:initial-creator>`	Asiakirjan ensimmäisen version luoja (**Yleistä > Luotu**)
`<meta:creation-date>`	Asiakirjan ensimmäisen version luontipäivämäärä ja -aika (**Yleistä > Luotu**)
`<meta:keyword>`	Avainsana (**Kuvaus > Avainsanat**). Ikkunassa pilkuin erotetut avainsanat esitetään erillisinä `meta:keyword`-kenttinä.
`<meta:editing-cycles>`	Muokkausten lukumäärä (**Yleistä > Muokkauskertoja**)
`<meta:editing-duration>`	Muokkauksen kokonaisaika (**Yleistä > Muokkauksen kokonaisaika**)
`<meta:user-defined>`	Käyttäjän määrittelemät metatiedot (**Käyttäjän määrittämä > Info 1**, jne.). Nimi annetaan kentässä

	`meta:name`.
`<meta:document-statistic>`	Tilastotietoja asiakirjasta (**Tilastotiedot**-välilehti). Elementillä on yli kymmenen tilastotietoattribuuttia.

6.6. Tyylit – styles.xml

Tiedostossa `styles.xml` määritellään kaikki käyttäjän muokkaamat sisäänrakennetut ja omat tyylit. Tiedoston juurielementtinä on `<office:document-styles>`, joka sisältää nimiavaruusmäärittelyt. Juurielementin alla voi olla seuraavia elementtejä:

`<office:font-face-decls>`	Fonttien esittelyjä, jotka sisältävät informaatiota asiakirjan kirjoittajan käyttämistä fonteista. Kaikkia näitä fontteja ei välttämättä käytetä kyseisessä asiakirjassa. Mikäli asiakirjassa käytettyä fonttia ei löydy toisesta tietokoneesta, jossa asiakirja avataan, auttavat nämä esittelyt toimisto-ohjelmistoa lähinnä vastaavan fontin löytämisessä.
`<office:styles>`	Sisäänrakennetut ja käyttäjän määrittelemät tyylit. Asiakirjan sisältämät tyylit riippuvat asiakirjan tyypistä. Ks. kunkin asiakirjatyypin kuvaus alla.
`<office:automatic-styles>`	Kohdekohtaisista muotoiluista automaattisesti luodut tyylit. Tehtäessä kappaleisiin, merkkeihin, tms. kohdekohtaisia asetuksia ohjelmisto luo niitä varten automaattisesti käyttäjälle näkymättömän tyylin, joka perustuu kohteen varsinaiseen tyyliin.
`<office:master-styles>`	Sisältää sivutyyleihin liittyvien ylä- ja alatunnisteiden sisällön.

Tyylit määritellään `<office:styles>`-elementin sisällä `<style:style>`-elementeissä. Tyylejä on monenlaisia ja eri asiakirjatyypit sisältävät erilaisten kohteiden asetuksia määrittteleviä tyylejä. Alla annetussa esimerkissä määritellään tässä raportissa käytetty leipätekstityyli. Tyylin määrityksessä on useita tärkeitä attribuutteja, jotka on kuvattu seuraavassa taulukossa.

`style:name`	Tyylin varsinainen nimi tunnisteena, joka ei voi sisältää muita kuin kirjaimia a-z, numeroita ja alaviivan. Siksi esimerkiksi välilyönti joudutaan koodaamaan alaviivoilla erotetulla heksadesimaalikoodilla. Automaattisten tyylien nimet generoidaan automaattisesti tallennettaessa, eikä voida luottaa, että ne pysyisivät samoina eri tallennuskerroilla. Näiden tyylien nimet koostuvat tyyliperheen mukaisesta kirjaimesta, esimerkiksi P kappaletyyleille, ja juoksevasta numerosta, esimerkiksi P1, P2, P3, jne.
`style:display-name`	Tyylin käyttäjälle näkyvä nimi. Sisäänrakennettujen tyylien nimi on OpenDocument-asiakirjassa aina englanniksi ja esimerkin ”Text body” näkyy suomenkielisessä ohjelmassa ”Leipäteksti”.

`style:family`	Tyylin perhe. Tyyli yksilöidään sen perheen ja nimen perusteella. Perheitä ovat asiakirjoissa käytettävien erilaisten kohteiden mukaisesti `paragraph`, `text`, `section`, `table`, `table-column`, `table-row`, `table-cell`, `table-page`, `chart`, `default`, `drawing-page`, `graphic`, `presentation`, `control` ja `ruby`.
`style:parent-style-name`	Tyylin perintähierarkian mukainen perustatyyli. Esimerkiksi kaikkien kappaletyylien ylin perustatyyli on Standard eli suomeksi Oletus.
`style:class`	Tyylin luokka.

Seuraavassa esimerkissä määritellään tässä raportissa käytetty Leipäteksti-kappaletyyli.

```
<!-- Leipätekstityylin määritys. Tyyli perustuu Oletus-tyyliin (Standard). -->
<style:style style:name="Text_20_body"
             style:display-name="Text body"
             style:family="paragraph"
             style:parent-style-name="Standard"
             style:class="text"
             style:master-page-name="">
  <!-- Kappaletyylissä määritellään erityisesti kappaleominaisuudet. -->
  <style:paragraph-properties  fo:margin-left="0cm"
                               fo:margin-right="0cm"
                               fo:margin-top="0cm"
                               fo:margin-bottom="0.439cm"
                               fo:widows="2"
                               fo:hyphenation-ladder-count="no-limit"
                               fo:text-indent="0cm"
                               style:auto-text-indent="false"
                               fo:background-color="transparent">
    <!-- Kappaleella voi olla myös taustakuva - tässä ei. -->
    <style:background-image/>
  </style:paragraph-properties>

  <!-- Kappaletyylin tekstiominaisuudet ovat erikseen. Tässä
       määritellään tavutusparametrit ja kytketään tavutus pois käytöstä. -->
  <style:text-properties  fo:hyphenate="false"
                          fo:hyphenation-remain-char-count="2"
                          fo:hyphenation-push-char-count="2"/>
</style:style>
```

Esimerkissä määritellyssä kappaletyylissä määritellään tyylin kappaleominaisuudet `<style:paragraph-properties>`-elementin attribuuteissa ja sisällä, sekä fonttimääritykset `<style:text-properties>`-elementissä.

Tyylimäärityksiä voidaan lisäksi tehdä `content.xml`-tiedostossa (ks. alla) automaattisten tyylien osalta.

6.7. Sisältö – content.xml

Oleellisin tiedostoista on `content.xml`, joka sisältää asiakirjan varsinaisen sisällön. Tiedoston ylimpänä elementtinä on `<office:document-content>`, jossa tehdään myös asiakirjan nimiavaruusmäärittelyt. Ennen varsinaista asiakirjasisältöä on joukko erilaisia määrityksiä, jotka luetellaan seuraavassa taulukossa.

`<office:scripts>`	Asiakirjan sisältämät makrot. OpenOfficessa yleensä

	StarBasic-kielellä.
`<office:font-face-decls>`	Fonttien esittelyjä, jotka sisältävät informaatiota asiakirjan kirjoittajan käyttämistä fonteista. Kaikkia näitä fontteja ei välttämättä käytetä kyseisessä asiakirjassa. Mikäli asiakirjassa käytettyä fonttia ei löydy toisesta tietokoneesta, jossa asiakirja avataan, auttavat nämä esittelyt toimisto-ohjelmistoa lähinnä vastaavan fontin löytämisessä.
`<office:automatic-styles>`	Automaattiset tyylit, ks. `styles.xml` ja `content.xml`.
`<office:body>`	Asiakirjan runko, joka sisältää asiakirjatyypin mukaisen elementin.

Esimerkiksi tekstiasiakirjojen `content.xml`-tiedoston yleisrakenne on seuraavanlainen:

```
<office:document-content>
   <office:scripts>
      ...asiakirjaan liitetyt makrot...
   </office:scripts>
   <office:font-face-decls>
      ...fonttimäärittelyjä...
   </office:font-face-decls>
   <office:automatic-styles>
      ...tyylimäärittelyjä...
   </office:automatic-styles>
   <office:body>
      <office:text>
         ...asiakirjan teksti ja rakennesisältö...
      </office:text>
   </office:body>
</office:document-content>
```

Asiakirjan alussa tehtävistä määrittelyistä ehkä tärkeimmät ovat `<office:font-face-decls>`, jossa määritellään asiakirjan sisältämät fontit, sekä `<office:automatic-styles>`, jossa määritellään kohdekohtaisia asetuksia varten automaattisesti luodut tyylit. Automaattiset tyylit pohjautuvat yleensä johonkin varsinaiseen tyyliin, jotka määritellään `styles.xml`-tiedostossa.

Tarkastelemme alla luvusta 6.9 eteenpäin `<office:body>`-elementin sisältöä erilaisissa asiakirjatyypeissä. Kuhunkin näihin liittyy myös erilaisia määrittelyjä tiedoston alussa, kuten tyylimääritykset.

6.8. Yhden XML-tiedoston muoto

OpenDocument-muodolla on ZIP-paketin lisäksi vaihtoehtoinen esitysmuoto yhtenä XML-tiedostona olevana asiakirjana. Tätä muotoa ei yleensä käytetä varsinaisena tallennusmuotona, koska tiedostot vievät huomattavasti tilaa ja niiden käsittely on hidasta. Muoto on kuitenkin merkittävä, sillä OpenOfficen XSLT-pohjaiset vientisuotimet saavat OpenDocument-asiakirjan syötteenään yhden XML-tiedoston muodossa ja vastaavasti tuontisuodin voi tuottaa asiakirjan tässä muodossa. Liitteessä C (s. 71) näytetään, kuinka tätä esitysmuotoa voidaan käyttää identiteettimuunnoksen avulla.

Yhden XML-tiedoston muodossa juurielementtinä on `<office:document>`. Sen alla ovat seuraavat elementit:

`<office:meta>`	Metatiedot, ks. `meta.xml`.
`<office:settings>`	Ohjelmakohtaiset asetukset, ks. `settings.xml`.
`<office:scripts>`	Makrot, ks. `content.xml`.
`<office:font-face-decls>`	Fonttimääritykset, ks. `styles.xml` ja `content.xml`.
`<office:styles>`	Tyylimääritykset, ks. `styles.xml`.
`<office:automatic-styles>`	Automaattiset tyylit, ks. `styles.xml` ja `content.xml`.
`<office:master-styles>`	Sivupohjatyylit, ks. `styles.xml`.
`<office:body>`	Asiakirjan runko.

Asiakirjan rungon `<office:body>` alla on asiakirjatyypistä riippuva asiakirjasisällön sisältävä elementti, kuten tekstiasiakirjoissa `<office:text>`. Tiedoston sisältö on tällöin seuraavanlainen:

```
<office:document ...nimiavaruusmäärittelyt...>
   <office:meta>
      ...meta.xml-tiedoston sisältöä...
   </office:meta>
   <office:settings>
      ...settings.xml-tiedoston sisältöä...
   </office:settings>
   <office:scripts>
      ...makroja...
   </office:scripts>
   <office:body>
      <office:text>
         ...asiakirjan sisältö...
      </office:text>
   </office:body>
</office:document>
```

6.9. Tekstiasiakirjat

Tekstiasiakirjan runko on `<office:text>`-elementin sisällä. Tekstiasiakirjat koostuvat ensisijaisesti kappaleista, jotka sisältävät tekstiä. Kappaleet esitetään `<text:p>`-elementeillä (*p* tulee englannin sanasta *paragraph*). Minimaalisen tekstiasiakirjan sisältö on, edellä mainittujen alkumääritysten jälkeen, jotakuinkin seuraavanlainen:

```
...
<office:body>
 <office:text>
  <office:forms form:automatic-focus="false" form:apply-design-mode="false"/>
  <text:sequence-decls>
   <text:sequence-decl text:display-outline-level="0" text:name="Illustration"/>
   <text:sequence-decl text:display-outline-level="0" text:name="Table"/>
   <text:sequence-decl text:display-outline-level="0" text:name="Text"/>
   <text:sequence-decl text:display-outline-level="0" text:name="Drawing"/>
  </text:sequence-decls>
  <text:p text:style-name="Standard">Tässä on kappale</text:p>
 </office:text>
</office:body>
```

Tekstiasiakirjan rungon alussa tehdään muun muassa asiakirjan sisältämien lomakkeiden määrittelyt `<office:forms>`-elementissä, sekä erilaisia esittelyitä. Monet elementit, kuten käyttäjän määrittämät muuttujakentät, vaativat tekstiasiakirjan alussa tehtävän esittelyn.

Oletuksena tehdään ainakin erilaisten numerointien, kuten kuvien, taulukoiden, tekstikehysten ja piirrosten numerointien esittelyt `<text:sequence-decls>`-elementissä.

Varsinaiset tekstikappaleet esitetään `<text:p>`-elementeissä. Elementissä määritellään aina kappaleen kappaletyyli `text:style-name`-attribuutilla. Mikäli kappaleeseen on tehty kappalekohtaisia muutoksia, viitataan automaattiseen tyyliin, joka määritellään `content.xml`-tiedoston alussa. Automaattinen tyyli perustuu kappaleen varsinaiseen tyyliin.

Kappaleen sisällä voi olla tekstijaksoja, jotka merkitään `<text:span>`-elementillä. Span-merkintä sisältää `text:style-name`-attribuutin, joka määrittää merkkityylin: joko tavallisen tai automaattisen. Esimerkiksi seuraavassa on kappale: ”Tässä *on* tekstiä.”

```
<text:p text:style-name="Standard">TässäĊ
  <text:span text:style-name="T1">on</text:span> tekstiä.</text:p>
```

Kappaleen sisältämä kursiivilla kirjoitettu *on*-sana on ympäröity `<text:span>`-elementillä, jonka merkkityyli on ”`T1`”. Tämä on automaattisesti luotu tyyli, joka on määritelty `content.xml`-tiedoston alussa seuraavasti:

```
<office:automatic-styles>
  <style:style style:name="T1" style:family="text">
    <style:text-properties  fo:font-style="italic"
                            style:font-style-asian="italic"
                            style:font-style-complex="italic"/>
  </style:style>
</office:automatic-styles>
```

Merkkien, kappaleiden ja muiden objektien asetukset määritellään OpenDocumentissa aina tyyleissä. Osa tyyleistä on vakiotyylejä, kuten Oletus tai Leipäteksti, osa käyttäjän määrittelemiä tyylejä ja osa automaattisia tyylejä. Automaattinen tyyli luodaan aina, kun asiakirjan muotoiluasetuksiin tehdään kohdekohtaisia, tyyleistä poikkeavia asetuksia. Esimerkiksi tekstinpätkän lihavointi aiheuttaa uuden merkkityylin luomisen. Tyylimäärittelyistä kerrotaan tarkemmin yllä `styles.xml`-tiedoston kuvauksen yhteydessä.

Otsikkokappaleita ei kirjoiteta `<text:p>`-elementillä vaan omalla `<text:h>`-elementillään. Tämän tärkein attribuutti on `text:outline-level`, joka määrittelee otsikon jäsennystason. Otsikon numerointi tehdään jäsennystason mukaan, käyttäen tavallisimmin jäsennysnumerointia (OpenOfficessa **Työkalut > Jäsennysnumerointi**), mutta mahdollisesti myös otsikkotyyliin sidotun numerointityylin määrittämää numerointia.

XML:ssä välilyöntien lukumäärällä ei ole merkitystä, joten mikäli tekstissä on useampia peräkkäisiä välilyöntejä, ilmaistaan ne erityisellä `<text:s>`-elementillä, jonka `text:c`-attribuutissa annetaan välilyöntien lukumäärä, esimerkiksi: `<text:s text:c="3"/>`. Vastaavasti sarkainmerkit ilmaistaan erityisellä `<text:tab/>`-elementillä.

Kappaleiden ja tekstin lisäksi tekstiasiakirjat voivat sisältää kuvia, grafiikkaobjekteja, tekstikehyksiä, ohjausobjekteja, kenttiä ja muita erikoisobjekteja. OpenDocument tukee myös joitain tekstiasiakirjojen ominaisuuksia, joita OpenOffice ei tue. Tällaisia ovat esimerkiksi sivusarjat, joita tarvitaan taitto-ohjelmien, kuten KOfficen KWordin vaatimiin ominaisuuksiin. Sivusarjoja käytettäessä asiakirja on täysin sivupohjainen, kuten OpenOffice Draw- tai Impress-asiakirjakin, ja kaikki tekstisisältö kirjoitetaan tekstikehyksiin.

6.10. Laskentataulukot

Taulukkolaskenta-asiakirjat koostuvat taulukkojen sarjasta. Laskentataulukot voivat lisäksi sisältää lomakkeita, muutosten seurantatietoa, sekä erilaisia määrityksiä, jotka kaikki määritellään joko rungon alkuosassa tai loppuosassa. Seuraavassa kommentoidussa esimerkissä määritellään yksinkertainen taulukko, jonka ensimmäisellä rivillä on kolme solua.

```
...
<office:body>
  <office:spreadsheet>

    <!-- Rungon alussa tehtäviä määrityksiä -->
    <table:calculation-settings>
      <!-- Asetetaan käytettävä päivämäärästandardi. -->
      <table:null-date table:date-value="1900-01-01"/>
    </table:calculation-settings>

    <!-- Rungon pääosan muodostavat taulukot -->
    <table:table table:name="Taulukko1" table:style-name="ta1">

      <!-- Taulukon alussa on sarakkeiden tyylimäärittely. -->
      <table:table-column  table:style-name="co1"
                           table:number-columns-repeated="9"
                           table:default-cell-style-name="Default"/>

      <!-- Taulukon rivi, joka koostuu soluista. -->
      <table:table-row table:style-name="ro1">

        <!-- Tekstisolu. Sisältö on text:p-elementissä. -->
        <table:table-cell office:value-type="string">
          <text:p>Yksi</text:p>
        </table:table-cell>

        <!-- Luvun sisältävä solu. Arvo ja teksti on tallennettu erikseen. -->
        <table:table-cell  office:value-type="float"
                           office:value="42">
          <text:p>42</text:p>
        </table:table-cell>

        <!-- Laskentasolu. Kaava, tulos ja tekstisisältö ovat erikseen. -->
        <table:table-cell  table:formula="oooc:=[.B1]/6"
                           office:value-type="float"
                           office:value="7">
          <text:p>7</text:p>
        </table:table-cell>

        <!-- Lopussa olevat tyhjät solut vain ohitetaan. -->
        <table:table-cell table:number-columns-repeated="6"/>
      </table:table-row>
      ...
    </table:table>
    ...
  </office:spreadsheet>
</office:body>
```

OpenDocumentin laskentataulukot ja tekstiasiakirjojen taulukot on toteutettu samojen elementtien avulla. Taulukkomalli on saman kaltainen kuin HTML:ssä eli taulukko koostuu riveistä, jotka koostuvat soluista. Eri riveillä voi olla eri määrä soluja, jolloin pois jätetyt solut ovat tyhjiä. Tyhjät solut voidaan ohittaa solun `table:number-columns-repeated`-attribuutilla. Taulukot voivat myös olla sisäkkäisiä.

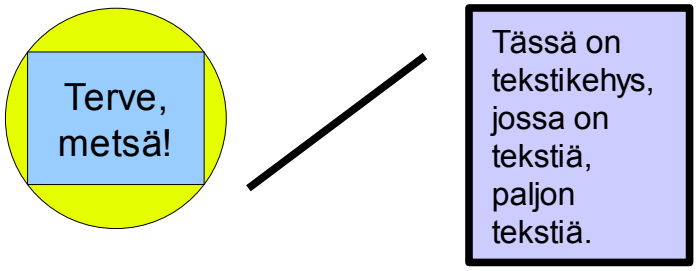
Laskentataulukkokaavoja ei OpenDocument-standardin versio 1.1 vielä määrittele. Kaavat ovat tulossa standardin 1.2-versioon.

Päivämäärien ja kellonaikojen esitys noudattaa ISO 8601 -standardia, jossa ne esitetään karkeasti ottaen muodossa ”VVVV-KK-PP hh:mm:ss”. Standardi sisältää myös muun muassa aikavyöhykkeiden, viikkonumeroiden, ajankestojen ja aikavälien merkintätavat. Huomion arvoista on, että taulukkolaskentaohjelmat suorittavat laskutoimituksia päivämäärillä ja ajoilla muunnettuaan ne ensin liukulukumuotoon. Tällöin laskenta tehdään pohjautuen johonkin kantapäivämäärään, kuten 30.12.1899. Kantapäivämäärä määritetään `<table:null-date>`-elementissä (ks. esimerkki yllä).

6.11. Piirrokset

Piirroksen runko on `<office:drawing>`-elementin sisällä. Rungon alussa voi olla tekstimäärittelyjä, mutta muuten se koostuu `<draw:page>`-elementillä merkityistä sivuista. Sivut pohjautuvat aina johonkin pohjasivuun, joka määritetään `draw:master-page-name`-attribuutilla. OpenOfficessa pohjasivuja voi muokata valikosta **Näytä > Pohja**. Sivun `draw:style-name`-attribuutissa kerrotaan sivun asetukset määrittävä sivutyyli, joka on aina automaattinen tyyli. Sivujen sisältö koostuu yleensä suurelta osin piirrosobjekteista, joilla on kohdekohtaisten sijaintiparametrien lisäksi grafiikkatyyli ja tekstisisällön tyylin määrittävä tekstityyli. Kaikki piirrosobjektit voivat sisältää tekstiä, joka määritellään piirrosobjektin elementin sisällä `<text:p>`-elementissä.

Tarkastellaan esimerkkinä seuraavaa piirrosta.



Piirros 1: Esimerkkipiirros.

Tämän neljä piirrosobjektia sisältävän piirrosasiakirjan runko on kommentoituna seuraavanlainen.

```
<office:body>
  <office:drawing>

    <!-- Piirrosasiakirja koostuu sivuista, jotka pohjautuvat pohjasivuun. -->
    <draw:page    draw:name="page1"
                  draw:style-name="dp1"
                  draw:master-page-name="Oletus">

      <!-- Sivun alussa oleva määritys sivulla olevista lomakkeista. -->
      <office:forms   form:automatic-focus="false"
                      form:apply-design-mode="false"/>

      <!-- Ellipsi-piirrosobjekti. -->
      <draw:ellipse   draw:style-name="gr1" draw:text-style-name="P1"
                      draw:layer="layout" svg:width="5cm" svg:height="4.991cm"
                      svg:x="2.5cm" svg:y="2.009cm">
        <!-- Ellipsi ei sisällä tekstiä, mutta tekstielementti on silti. -->
        <text:p text:style-name="P3"/>
      </draw:ellipse>
```

```
        <!-- Suorakaide, joka määritellään kulmapisteen, leveyden ja
             korkeuden perusteella. -->
        <draw:rect   draw:style-name="gr2" draw:text-style-name="P1"
                     draw:layer="layout" svg:width="4cm" svg:height="3cm"
                     svg:x="3cm" svg:y="3cm">
          <!-- Tässä piirrosobjektissa on kaksi riviä (kappaletta) tekstiä. -->
          <text:p text:style-name="P2">Terve,</text:p>
          <text:p text:style-name="P2">metsä!</text:p>
        </draw:rect>

        <!-- Suora, joka määritellään alku- ja loppupisteensä perusteella. -->
        <draw:line   draw:style-name="gr3" draw:text-style-name="P1"
                     draw:layer="layout" svg:x1="12cm" svg:y1="3.09cm"
                     svg:x2="8cm" svg:y2="6.09cm">
          <!-- Suora ei sisällä tekstiä. -->
          <text:p text:style-name="P3"/>
        </draw:line>

        <!-- Tekstikehys, joka määritellään paljolti kuten suorakaide. -->
        <draw:frame  draw:style-name="gr4" draw:text-style-name="P3"
                     draw:layer="layout" svg:width="5cm" svg:height="5cm"
                     svg:x="13cm" svg:y="2cm">
          <!-- Kehys sisältää tekstiruudun, joka sisältää tekstisisällön. -->
          <draw:text-box>
            <text:p text:style-name="P3">Tässä on tekstikehys, jossa on
                                         tekstiä, paljon tekstiä.</text:p>
          </draw:text-box>
        </draw:frame>
      </draw:page>
    </office:drawing>
</office:body>
```

Kohteiden sijainti määritellään `svg:x`- ja `svg:y`-attribuuteissa koordinaatteina sivun vasemmasta ylänurkasta. Grafiikkaobjektit sijaitsevat aina jossain kerroksessa, joka määritetään `draw:layer`-attribuutissa.

6.12. Esitykset

Esitykset ovat perusrakenteeltaan käytännössä identtisiä piirrosten kanssa ja niiden runko koostuu pääasiassa `<draw:page>`-elementeistä. Kuten piirroksissa, sivut pohjautuvat aina pohjasivuun, joka määritetään `draw:master-page-name`-attribuutissa. Kun piirroksissa ei pohjasivuja yleensä käytetä lainkaan, on niillä esityksissä yleensä keskeinen merkitys.

6.13. Tietokannat

OpenDocument ei määrittele tietokantayhteyksiä tai tietokantoja, eikä varsinkaan tietokantojen esitysmuotoa. Esimerkiksi OpenOffice Base -sovelluksen käyttämä tiedostomuoto ei, toisin kuin muiden OpenOffice-sovellusten, sisälly OpenDocument-standardiin. Sen tietokantayhteyksien määrityksiin ja HSQLDB-tietokantojen tallennukseen käyttämä .odb-tiedostomuoto on tosin hyvin pitkälti yhtenevä OpenDocument-asiakirjamuotojen kanssa. On odotettavissa, että tämä tiedostomuoto liitetään myöhemmin osaksi OpenDocument-standardia.

OpenDocumentiin sisältyy monia tietokantaliittymien vaatimia ominaisuuksia, kuten tietokantakenttiä ja lomakkeiden ohjausobjekteja, jotka voidaan kytkeä tietokantoihin. Toisaalta tietokantayhteys voidaan muodostaa vaikkapa OpenDocument-laskentataulukkoon.

Seuraavassa esimerkissä viitataan tietokantakentällä tietokantaan.

```
<text:database-display
   text:table-name="Taulu1"
   text:table-type="table"
   text:column-name="Etunimi"
   text:database-name="osoitelistaesimerkki">Kaarle</text:database-display>
```

Attribuutilla `text:database-name` määritellään tietokanta, attribuutilla `text:table-name` taulun nimi ja attribuutilla `text:column-name` sarakkeen nimi. Taulun nimi voi itse asiassa viitata joko varsinaiseen tauluun tai kyselyn tai SQL-komennon tulokseen; tyyppi määritellään `text:table-type`-attribuutilla, joka voi saada arvot `table`, `query` tai `command`. Elementin sisällä on tietokannasta haettu teksti, ylläolevassa esimerkissä arvo ”Kaarle”. Mikäli sisältöä ei vielä ole haettu, näytetään elementissä esimerkiksi sarakkeen nimi.

Toinen viittaustapa tietokantojen sisältöön ovat lomakkeiden ohjausobjektit. Niissä voidaan paitsi näyttää tietokantojen sisältöä, myös syöttää ja muokata sitä.

6.14. OpenDocumentin rajoitukset

OpenDocument ei määrittele asiakirjojen kaikkia yksityiskohtia. Toteutuksesta riippuvia tekijöitä ovat muun muassa seuraavat:

- Fontit
- Fonttien piirtoalgoritmit
- Laskentataulukkojen funktiot (tulossa ODF 1.2:ssa)
- Makrokieli
- Tavutusalgoritmit
- Rivitys- ja sivutusalgoritmit

Käytetyt fontit vaikuttavat muotoiluun merkittävästi. Kirjaimet ovat eri fonteissa eri kokoisia, jolloin sanoista tulee eri mittaisia ja ohjelma saattaa rivittää ne eri tavalla. Asiakirjoissa käytettävissä olevat fontit riippuvat siitä, mitä fontteja käyttöjärjestelmään tai itse toimisto-ohjelmistoon on asennettu. Mikäli fonttia ei ole saatavilla, voidaan se korvata toisella, samankaltaisella fontilla, jonka mitoissa saattaa kuitenkin olla pieniä eroja. OpenDocument ei toistaiseksi tue fonttien sisällyttämistä asiakirjoihin. Sisällyttäminen saattaisi olla ongelmallista muun muassa tekijänoikeussyistä – monissa maissa fontit ovat tekijänoikeussuojan alaisia ja fontin sisällyttäminen asiakirjaan voitaisiin tulkita sen kopioinniksi. Toisaalta muun muassa PDF tukee fonttien sisällyttämistä.

Samatkin fontit saatetaan piirtää eri ohjelmissa ja käyttöjärjestelmissä eri tavoilla. Piirtotapa voi vaikuttaa paitsi ulkoasuun, myös fonttien mittoihin, jolloin rivitys voi muuttua. Esimerkiksi tavalliset vektoripohjaiset TrueType-fontit saatetaan piirtää erilaisilla rasterointialgoritmeilla. Erityisesti fonttien pehmennys eli antialiasointi saatetaan tehdä eri algoritmeilla. Yhteensopivuutta saattavat vaikeuttaa myös rasterointialgoritmien patentit. Näin on esimerkiksi tiettyjen TrueType-piirtovihjeiden tapauksessa, joiden käsittelyalgoritmit ovat Applen patentoimia. Tästä syystä esimerkiksi Linuxissa käytetty fonttien rasterointikirjasto Freetype ei oletuksena tue kaikkia piirtovihjeitä. Vastaavasti Microsoftilla on hallussaan ClearType-fontteja koskeva patentti. UNIX- ja Linux-järjestelmissä fontin piirtoon vaikuttaa myös näyttömetriikka, jonka avulla fontti pyritään saamaan näkymään samanlaisena erilaisilla näytöillä.

OpenOfficessa on mahdollista käyttää myös tulostimen fonttimetriikkaa. Se poikkeaa yleensä näytön fonttimetriikasta ja on myös erilainen eri tulostimissa.

Toinen yleinen ongelma on tavutus. Esimerkiksi OpenOfficessa tavutus voi käyttää ohjelmaan sisäänrakennettuja tai laajennuksina asennettuja tavutuskomponentteja. Suomen kielen tavutusta varten on olemassa kolme vaihtoehtoa: sääntöpohjainen tavutuskomponentti, sekä sanastopohjaiset komponentit Soikko ja Voikko. Sanastopohjainen tavuttaja kykenee huomioimaan esimerkiksi yhdyssanat paremmin. Kaikki nämä kolme tavutuskomponenttia tavuttavat sanoja hieman eri tavoin, jolloin sanat jakautuvat riveille ja sivuille hieman eri tavalla. Tavutuskomponentti on myös saatettu jättää asentamatta käyttäjän tietokoneeseen ja joillekin kielille sellaista ei välttämättä ole lainkaan saatavilla.

Myös makrot saattavat olla ongelma eri OpenDocumentia tukevien ohjelmistojen välillä, sillä OpenDocument ei määrittele asiakirjoihin sisällytettyjen makrojen kieltä. OpenOfficessa käytetään makrokielenä StarBasic-kieltä. Lisäksi StarBasic-makroissa käytetään usein OpenOfficen UNO-rajapintaa, joka ei ole yleinen standardi.

7. SUOTIMET JA ASIAKIRJAMUUNNOKSET

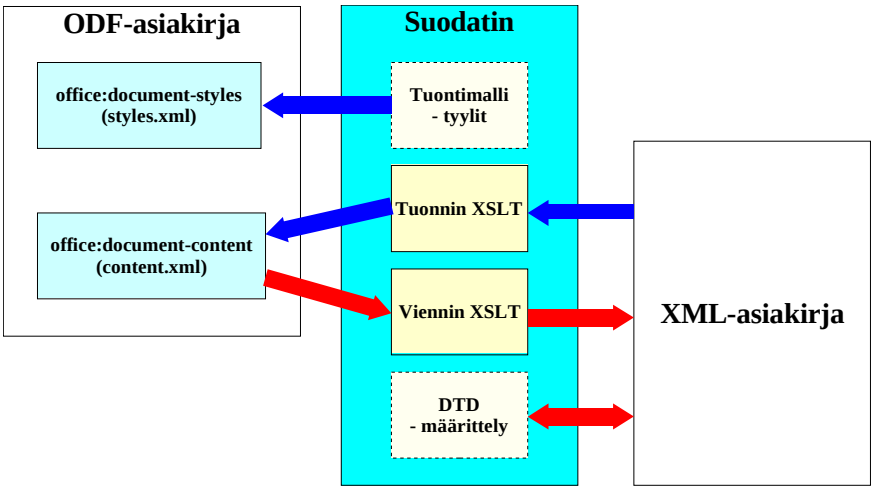
Tässä luvussa käsitellään OpenDocument-muodosta tai -muotoon tehtäviä asiakirjamuunnoksia yleisellä tasolla. Asiakirjamuunnoksia voidaan käyttää paitsi asiakirjojen muuntamiseen toisiin asiakirjamuotoihin, myös tietojen tai metatietojen keräämiseen asiakirjoista, sekä tietokantaraportti-asiakirjojen luomiseen. Muunnos voi poimia vain tiettyyn tarkoitukseen tarvittavia tietoja ja toisaalta muunnoksen avulla voidaan luoda hyvinkin suuri asiakirja yksinkertaistenkin syöteparametrien perusteella.

Muunnoksia voidaan tehdä kahdella eri tavalla: toimisto-ohjelmistoon asennetuilla suotimilla ja erillisillä ohjelmilla. Näitä kahta tapaa käsitellään alla tarkemmin. Tässä raportissa käsitellään erityisesti OpenOfficen suotimia; muissa OpenDocumentia tukevissa toimisto-ohjelmistoissa saatetaan käyttää toisenlaisia tekniikoita suotimien rakentamiseen.

7.1. XSLT-pohjaiset suotimet

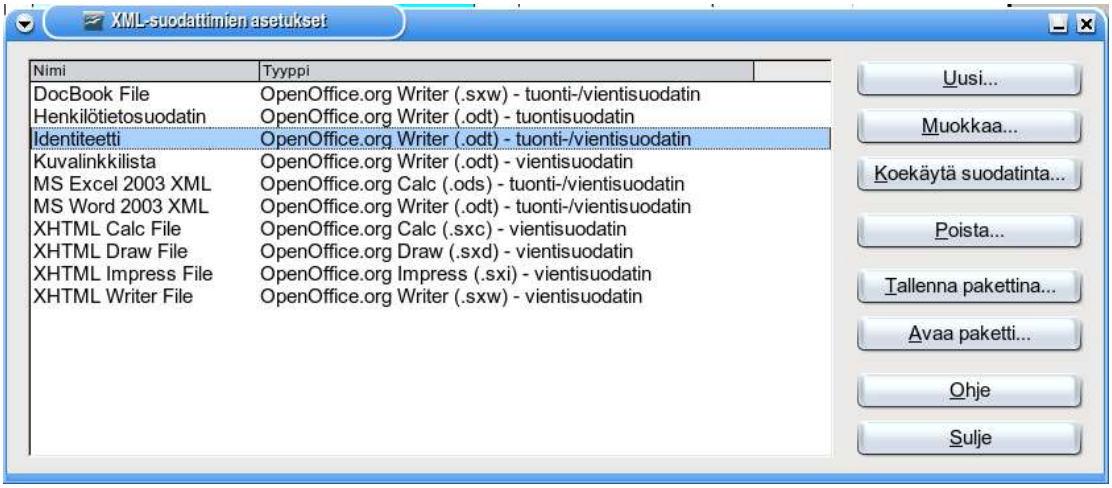
XSLT-muunnoskieli tarjoaa helpoimman tavan kehittää muun muassa OpenOfficessa tai ulkoisissa muunnosohjelmissa käytettäviä suotimia. Sen avulla tietyn muotoinen XML-asiakirja voidaan muuntaa toiseksi XML-asiakirjaksi tai johonkin muuhun muotoon, kuten HTML:ksi tai muotoilemattomaksi tekstiksi. XSLT-kielestä ja sen käsittelytyökaluista annetaan johdanto liitteessä B (s. 67).

OpenOfficen XSLT-suodin voi sisältää joko tuonti- tai vientisuotimen tai molemmat. Kaksisuuntainen suodin koostuu tuonnin ja viennin XSLT-muuntimista, sekä tuontimallista, joka sisältää asiakirjan tyylit. Vientisuotimelle on mahdollista määritellä DTD-rakennemäärittely (ks. Liite D), jota käyttäen voidaan tarkistaa, että viennin lopputuloksena oleva XML-tiedosto on oikean muotoinen.



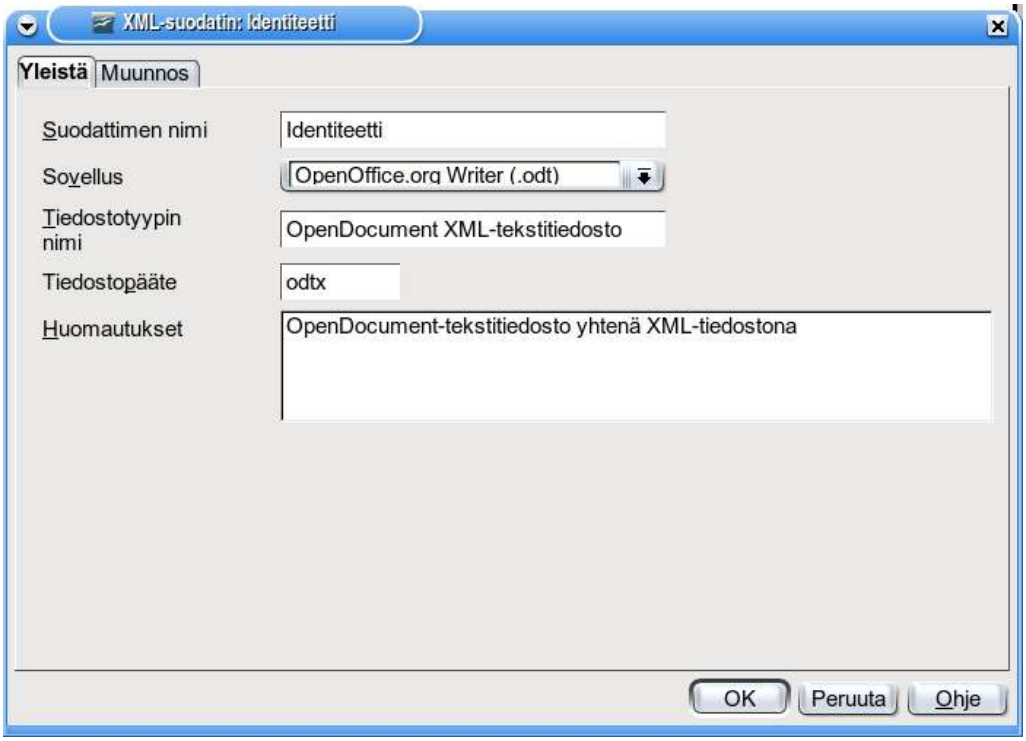
Kuva 8: Tuonti- ja vientisuotimet OpenOfficessa.

Kuva 8 kuvaa tuonti- ja vientisuotimien toimintaperiaatetta ZIP-muotoisilla OpenDocument-asiakirjoilla. XSLT-vientisuodin kuitenkin saa asiakirjan syötteenään yhden XML-tiedoston muodossa ja samoin tuontisuodin voi tuottaa koko asiakirjan luomalla juurielementiksi `<office:document>`-elementin. Vienti- ja tuontisuotimia käsitellään tarkemmin luvuissa 8 ja 9.



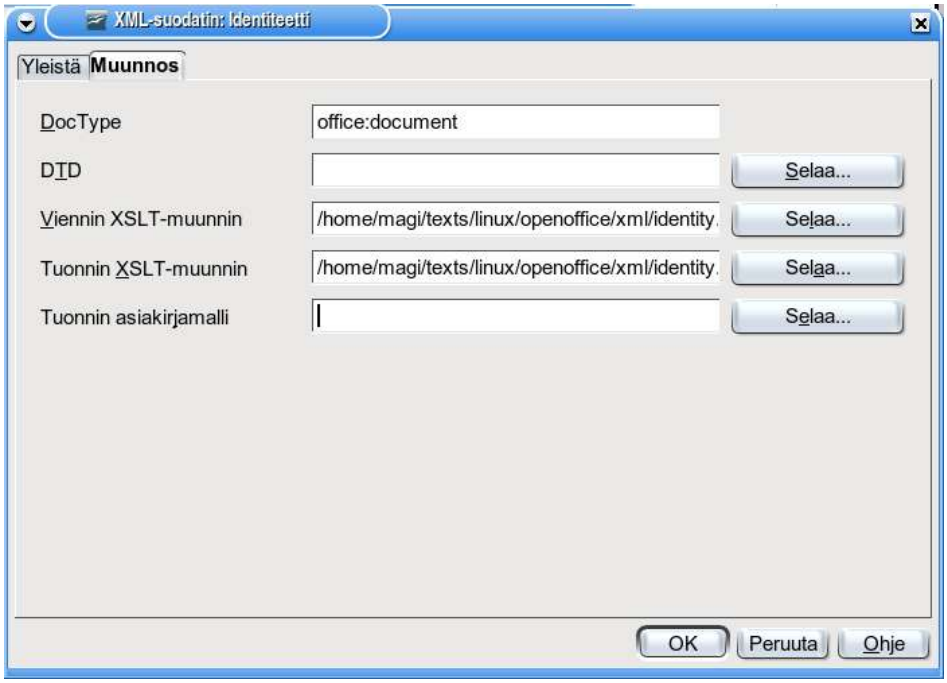
*Kuva 9: Valikosta **Työkalut > XML-suodattimien asetukset** avautuva asetusikkuna.*

XSLT-pohjaisia suotimia hallitaan OpenOfficessa valikosta **Työkalut > XML-suodattimien asetukset** (kuva 9). Ikkunasta voi luoda uuden suotimen **Uusi**-painikkeesta ja muokata olemassa olevaa **Muokkaa**-painikkeesta.



*Kuva 10: XML-suodattimen asetusikkunan **Yleistä**-välilehti.*

Suotimen asetusikkuna on esitetty kuvassa 10. Ikkunan **Yleistä**-välilehdellä annetaan **Suodattimen nimi** -kentässä nimi, joka näkyy **XML-suodattimien asetukset** -ikkunassa. **Sovellus**-kentässä määritellään asiakirjatyyppi, josta tai johon suodatin tekee muunnoksia. Vaihtoehtoina ovat OpenDocument-tyypit (esim. `.odt`) ja vanhat OpenOffice 1 -asiakirjamuodon tyypit (esim. `.sxw`). **Tiedostotyypin nimi** -asetusta käytetään käytettäessä suodinta **Tiedosto > Avaa**- tai **Tiedosto > Tallenna** -ikkunoissa, jolloin tiedostotyyppi näkyy muiden tiedostotyyppien listassa. **Tiedostopääte**-kentässä määritellään tiedostonimen tarkenne, jota käytetään avattaessa ja tallennettaessa tiedosto suodattimella avaus- ja tallennusikkunoissa. **Huomautukset** ovat vapaavalintaista tekstiä.



*Kuva 11: XML-suodattimen asetusikkunan **Muunnos**-välilehti.*

Muunnos-välilehdellä määritellään **DocType**-parametri. Tuontisuotimissa tätä käytetään asiakirjatyypin määrittämiseen. Jos määritelty **DocType**-parametri esiintyy tiedoston alussa, katsotaan asiakirja kyseiselle tuontisuotimelle kuuluvaksi ja avataan sen avulla. Esimerkiksi yllä kuvassa 11 määritellään identiteettituontisuotimen asiakirjatyypiksi office:document, joka on yhden XML-tiedoston muodossa olevan OpenDocument-asiakirjan juurielementti. Mikäli **DTD**-kenttään on annettu DTD-rakennemäärittelyn sisältävä tiedosto, käytetään sitä viennissä tulosteen XML-rakenteen tarkistamiseen. Tarkistusta ei voida tehdä, mikäli viennin tulos ei ole XML-muotoinen. **Viennin XSLT-muunnin** ja **Tuonnin XSLT-muunnin** ovat viittaukset viennin- ja tuonnin XSLT-muuntimiin. OpenDocument-asiakirja **Tuonnin asiakirjamalli** sisältää asiakirjan tyylit; tiedostosta luetaan vain `styles.xml`-tiedosto. Tällä on merkitystä vain jos tuontisuodin tulostaa vain `content.xml`-tiedoston sisältämän asiakirjan rungon. Mikäli tuontisuodin tuottaa koko asiakirjan yhden XML-tiedoston muodossa, jossa juurielementtina on `<office:document>` ja joka sisältää `<office:styles>`-elementin, ei malliasiakirjaa tarvita. Esimerkiksi liitteessä C esitetty identiteettisuodin tekee näin.

7.2. Suotimien ohjelmointi sovellusrajapintojen avulla

OpenOfficen suotimet on mahdollista toteuttaa myös muilla ohjelmointikielillä kuin XSLT:llä sovellusrajapintojen avulla. Suotimien ohjelmointi tällä tekniikalla on ilmaisuvoimaisin ja suorituskyvyltään tehokkain tapa, mutta työlästä verrattuna XML-pohjaisiin suotimiin. Kovin monimutkaisia suotimia, varsinkaan binääristen asiakirjamuotojen välillä, ei ole käytännössä mahdollista tehdä XSLT-muunnoksilla, jolloin sovellusrajapintojen käyttö on välttämätöntä.

Sovellusrajapintoja käyttävät suotimet asennetaan OpenOfficeen lisäosina valikosta **Työkalut > Lisäosien hallinta** tai ulkoisella asennusohjelmalla, joka asentaa ja rekisteröi komponentin.

Sovellusrajapintoja voidaan käyttää muunnosten tekemiseen kolmella tasolla: käyttäen OpenOfficen ydinrajapintaa, OpenOffice API -rajapintaa tai XML-tiedostorajapintaa.

OpenOfficen sisäinen rajapinta on tehokkain tapa tehdä suotimia. Niiden kehitys tapahtuu

aina C++-kielellä. OpenOfficen sisäänrakennetut suotimet on rakennettu tällä menetelmällä. Sisäisen rajapinnan käyttö on kuitenkin hankalaa, koska se saattaa muuttua hyvinkin helposti eri ohjelmaversioiden välillä. Käytännössä tällaiset suotimet täytyy linkata aina tiettyä ohjelmaversiota vasten, jolloin suotimet täytyy kääntää uudestaan jokaiselle uudelle OpenOffice-versiolle, mikä tekee niiden ylläpidosta työlästä.

UNO-pohjainen OpenOffice API -rajapinta on huomattavasti vakaammin määritelty kuin sisäinen rajapinta, joskin samalla hivenen tehottomampi. UNO-rajapintaa voidaan käyttää monista eri ohjelmointikielistä; valitsemalla toteutuskieli sopivasti voidaan löytää kompromissi tehokkuuden ja yhteensopivuuden välillä.

Kolmas ja kenties helppokäyttöisin vaihtoehto on UNO-pohjainen XML-tiedostorajapinta, joka saa syötteekseen asiakirjan sisällön XML-muodossa, aivan kuten XSLT-pohjaisetkin suotimet. Koska OpenOffice-asiakirja täytyy ensin muuntaa sisäisestä esityksestä XML-muotoon, on tämä tapa jonkin verran muita hitaampi.

Suotimien rakentamisesta sovellusrajapintoja käyttäen löytyy ohjesivusto osoitteesta:

http://xml.openoffice.org/filter/

7.3. Suodinpaketit

Yksi tai useampia XSLT-suotimia voidaan paketoida suodinpaketiksi. Suodinpaketti luodaan valikosta **Työkalut > XML-suodattimien asetukset** (kuva 9) valitsemalla pakettiin halutut suodattimet. Saat valittua useampia suodattimia pitämällä vaihtonäppäin tai Ctrl-näppäin painettuna suodattimia napsautettaessa. Kun olet valinnut halutut suodattimet, napsauta **Tallenna pakettina** ja anna paketin nimi. Tiedostotarkenteeksi tulee `.jar` (Java Archive).

Tallennettu suodinpaketti asennetaan samasta XML-suodattimien asetusikkunasta napsauttamalla **Avaa paketti** ja valitsemalla paketin tiedoston.

7.4. Ulkoiset työkalut

OpenOfficen suotimien lisäksi OpenDocument-asiakirjoista voidaan tehdä muunnoksia paitsi toisissa toimisto-ohjelmistoissa myös erillisillä ohjelmilla. XSLT-muunnoksia on mahdollista tehdä erillisillä XSLT-suorittimilla. Liitteessä B annetaan yleiskatsaus eri XSLT-suorittimista ja niiden käytöstä OpenDocument-asiakirjojen käsittelyssä. Muiden ohjelmointikielten käytöstä kerrotaan tarkemmin alla luvussa 10.

8. VIENTISUOTIMET

OpenOfficen vientisuotimet muuntavat OpenDocument-asiakirjan johonkin muuhun muotoon tallennettavaksi. Suotimet tehdään yleensä XSLT-muunnoskielellä, mutta monimutkaisissa tapauksissa voi olla tarpeellista käyttää muuta kieltä (ks. luku 7.2). XSLT-kielestä ja sen käsittelytyökaluista annetaan johdanto liitteessä B.

Vientisuotimia voidaan käyttää joko OpenOfficen sisäänrakennetulla toiminnolla tai ulkoisessa järjestelmässä ulkoisen XSLT-suorittimen avulla.

8.1. Vientisuodinten rakentaminen

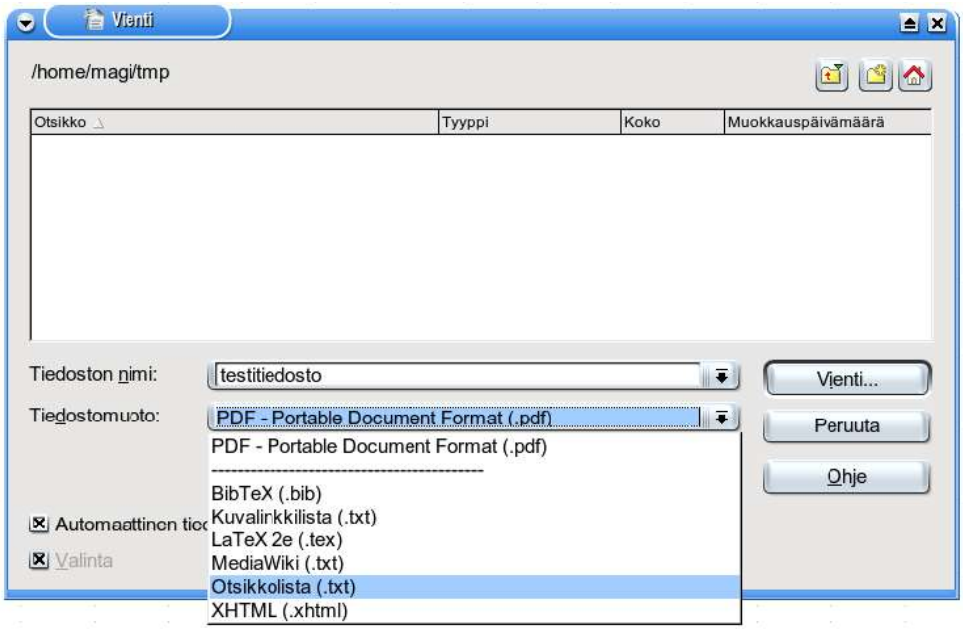
OpenOfficessa käytettävä XSLT-muunnin saa OpenDocument-asiakirjan käsiteltäväkseen yhden XML-tiedoston muodossa (ks. luku 6.8). Juurielementtinä on siten `<office:document>`. Muunnoksen tulos kirjoitetaan tiedostoon, joka voi olla XML-, HTML- tai muu tiedosto, sen mukaan mitä muunnoksen `<xsl:output>`-elementissä on määritelty. Tulostiedoston tarkenne määritellään XSLT-suotimen asetuksissa (ks. luku 7.1 edellä).

Vaikka OpenOffice ei itse asiassa tue lainkaan asiakirjojen tallentamista yhden tiedoston XML-muodossa, voidaan se toteuttaa identiteettimuunnoksella. Identiteettimuunnos ja sen asennus suotimeksi on kuvattu liitteessä C. Identiteettimuunnosta voi kannattaa käyttää XSLT-suotimien kehityksen aikana asiakirjojen XML-rakenteen ymmärtämisessä.

Vientisuotimelle voidaan lisäksi määritellä suotimen asetuksissa valinnainen DTD-rakennemäärittelytiedosto. Mikäli DTD on saatavilla, käytetään sitä viennin tulostiedoston tarkistamiseen. Mikäli tulostiedosto ei vastaa rakennemäärittelyä, vienti epäonnistuu. Rakennemäärittelyn käyttö voi olla hyödyllistä suotimen kehitysvaiheessa ja myöhemmin vikojen jäljittämisessä.

8.2. Vientisuotimien käyttö OpenOfficessa

Vientisuodinten asetukset OpenOfficessa on esitetty edellä luvussa 7.1. Kun suodin on asennettu, näkyy se automaattisesti tiedoston vienti-ikkunassa, kuten esitetty kuvassa 12. Viennissä käytetään oletuksena suotimen asetuksissa käytettyä tiedostotarkennetta.



Kuva 12: Tiedostomuodon valinta viennin tiedostovalintaikkunassa.

8.3. Vientisuotimet ulkoisessa järjestelmässä

OpenOfficelle tehtyä XSLT-suodinta voi käyttää yleensä suoraan sellaisenaan ulkoisissa järjestelmissä. Ainoana erona on, että kun OpenOfficen sisäinen vientisuodin saa OpenDocument-asiakirjan yhden XML-tiedoston muodossa (ks. luku 6.8), täytyy ulkoisessa järjestelmässä käsitellä asiakirjaa erillisinä XML-tiedostoina. Tämä edellyttää ZIP-pakatun OpenDocument-tiedoston purkua ennen käsittelyä.

Ulkoisessa järjestelmässä tehtävää XSLT-suodinta käyttävää muunnosta varten tarvitaan XSLT-suoritin. XSLT-suorittimista ja niiden käytöstä kerrotaan yleisemmin liitteessä B.2. ZIP-tiedostojen käsittelyyn on olemassa lukuisia työkaluja ja ohjelmointikirjastoja. Alla on esitetty, miten muunnoksia voidaan tehdä Linux-komentoriviltä käyttäen Info-ZIP-ohjelmaa ja Xalan-XSLT-suoritinta.

OpenDocument-tiedoston voi purkaa seuraavanlaisesti `unzip`-ohjelmalla:

```
$ unzip testi.odt
Archive:  ../testi.odt
 extracting: mimetype
  inflating: content.xml
  inflating: styles.xml
  inflating: meta.xml
  inflating: Thumbnails/thumbnail.png
  inflating: settings.xml
  inflating: META-INF/manifest.xml
```

Tämä purkaa koko OpenDocument-asiakirjan sisällön nykyiseen hakemistoon. Mikäli halutaan purkaa vain tietyt tiedostot, annetaan niiden nimet unzip-ohjelmalle paketin nimen jälkeen välilyönnillä erotettuna. Mikäli asiakirjasta halutaan käsitellä vain yhtä tiedostoa, esimerkiksi `content.xml`-tiedostoa, kannattaa se tulostaa `-p` -valitsimella suoraan oletustulosteeseen seuraavasti:

```
$ unzip -p testi.odt content.xml
<?xml version="1.0" encoding="UTF-8"?>
```

```
<office:document-content ...
```

Xalan-suoritin ottaa syötteensä oletuksena oletussyötteestä, jolloin voimme ohjata ZIP-purkuohjelman tulosteen suorittimen syötteeksi putkella seuraavasti:

```
$ unzip -p testi.odt content.xml | xalan -xsl otsikot.xsl
Reading input document from stdin...
ASIAKIRJAN OTSIKOT
* JOHDANTO
* AVOIN TIEDOSTOMUOTO
* OPENDOCUMENT-STANDARDIN KEHITTYMINEN
```

Käytimme yllä yksinkertaista vientisuodinta, joka tulostaa asiakirjan otsikot tekstitiedostoksi. Esitämme tämän suotimen toiminnan seuraavaksi.

8.4. Esimerkki 1: tekstiasiakirjan otsikoiden tulostaminen

Asiakirjan sisällysluettelo on eräänlainen metatieto, jota voidaan käyttää asiakirjan indeksoinnissa. Seuraavassa esimerkissä teemme muunnoksen tekstiasiakirjasta muotoilemattomaksi tekstitiedostoksi. Muunnoksessa tulostetaan asiakirjan kaikki otsikot jäsennystason mukaisesti sisennettyinä.

Suotimen XSLT-muunnin on seuraavanlainen:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!-- ======================================================================= -->
<!-- Juurielementti ja OpenDocument-nimiavaruusmäärittelyt                   -->
<!-- ======================================================================= -->
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:office="urn:oasis:names:tc:opendocument:xmlns:office:1.0"
  xmlns:text="urn:oasis:names:tc:opendocument:xmlns:text:1.0"
  office:version="1.0"
  version="1.0">

  <!-- Määrätään muunnoksen ulostulomuoto ja merkkikoodaus. -->
  <xsl:output method="text" encoding="ISO-8859-1"/>

  <!-- ======================================================================= -->
  <!-- Tee muotoilu otsikon text:level-kentässä olevan numeron mukaan  -->
  <!-- ======================================================================= -->
  <xsl:template match="text:h">
    <xsl:if test="normalize-space(.) != ''">

      <!-- Tulostetaan sisennys attribuutin text:outline-level mukaan. -->
      <xsl:value-of select="substring('          ', 1,
                           (number(@text:outline-level)-1)*2)"/>

      <!-- Tulostetaan *-merkki, otsikon teksti ja rivinvaihto. -->
      <xsl:text>* </xsl:text>
      <xsl:value-of select="."/>
      <xsl:text>&#x0a;</xsl:text>
    </xsl:if>
  </xsl:template>

  <!-- ======================================================================= -->
  <!-- Juurisolmu                                                              -->
  <!-- ======================================================================= -->
   <xsl:template match="/">
     <xsl:text>ASIAKIRJAN OTSIKOT&#x0a;</xsl:text>
     <xsl:apply-templates/>
   </xsl:template>
```

```
  <!-- ========================================================== -->
  <!-- Karsitaan pois kaikki muut tekstit                         -->
  <!-- ========================================================== -->
   <xsl:template match="text()">
   </xsl:template>
</xsl:stylesheet>
```

Huomaa, että muunnoksessa tehtiin tulosteeseen tulevat rivivaihdot UNIX-rivinvaihdolla eli pelkällä `
` (LF) -merkillä. Windows-käyttöjärjestelmässä tarvitaan `
` (CR+LF).

Kun sovellamme vientisuodinta tämän raportin OpenDocument-asiakirjaan, kuten edellä on kuvattu, saamme seuraavanlaisen tulosteen:

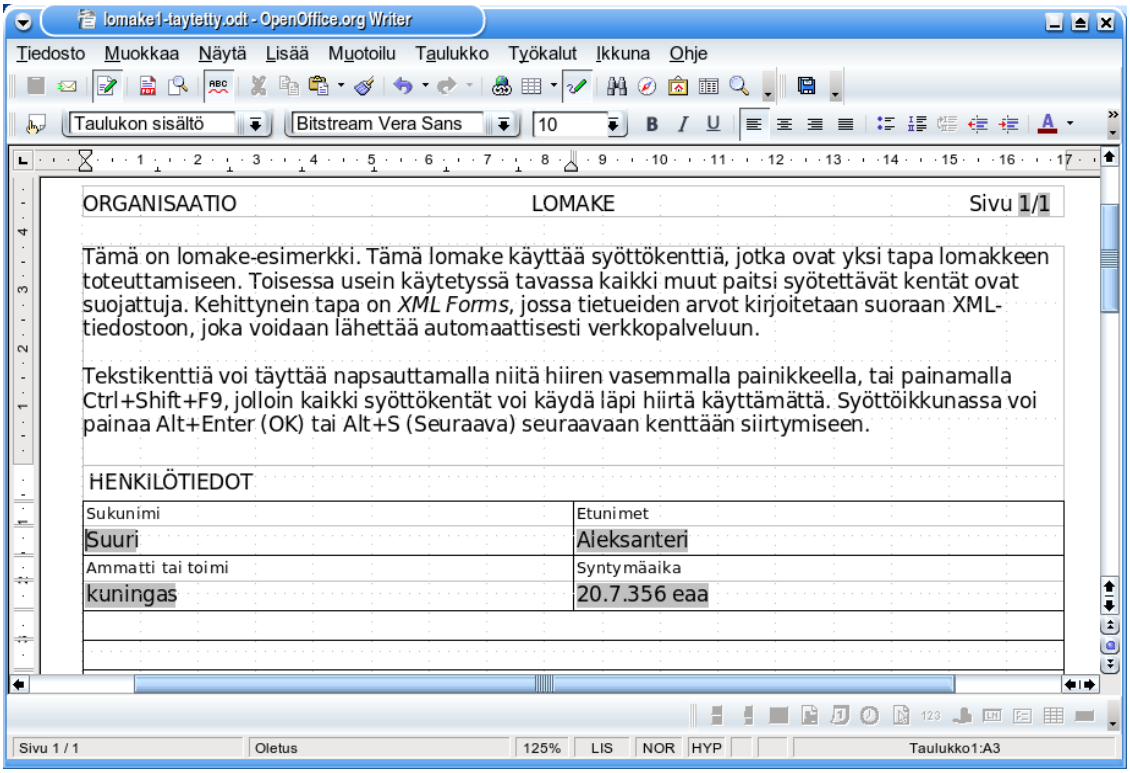
```
ASIAKIRJAN OTSIKOT
* JOHDANTO
* AVOIN TIEDOSTOMUOTO
* OPENDOCUMENT-STANDARDIN KEHITTYMINEN
  * OpenDocument-standardin käyttöönottotietoja
* EU-SUOSITUKSET AVOIMISTA TIEDOSTOMUODOISTA
* KÄYTTÖTAPAUKSIA JA RATKAISUJA
  * Toimisto-ohjelmisto osana työnkulkua
  * Integrointi sovellusrajapinnan avulla
```

8.5. Esimerkki 2: lomakkeen syöttökenttien kerääminen

Lomakkeiden sisällön lukemista tarvitaan tapauksissa, joissa käyttäjät palauttavat saamansa sähköisen lomakkeen tiedostona, josta lomakkeen tiedot luetaan automatisoidusti tietojärjestelmään. Tätä OpenDocument-tiedoston käyttötapausta on käsitelty edellä luvussa 5.9, jossa se muodostaa 3. sähköisten lomakkeiden käsittelytavan.

Syöttökentät ovat alkeellinen, mutta usein käytetty tapa toteuttaa sähköisiä lomakkeita. Syöttökentät ovat olleet yleisiä erityisesti Microsoft Word -ohjelmalla toteutetuissa lomakkeissa. Tyypillisesti käyttäjä napsauttaa syöttökenttiä yksitellen syöttääkseen niihin arvon. Mikäli lomake palautetaan tiedostona ja palautettavia tiedostoja on paljon, voi olla kannattavaa tehdä lomaketietojen tallennusjärjestelmään muunnin, joka poimii lomakkeista syöttökenttien arvot sen sijaan, että lomakkeet käsiteltäisiin manuaalisesti.

Syöttökenttälomakkeen syöttökentät lisätään OpenOfficessa valikosta **Lisää > Kentät > Muu** (**Ctrl+F2**), **Toiminnot**-välilehdeltä. Valitse tyypiksi **Syöttökenttä**, anna kentän viite eli nimi ja napsauta **Lisää**. Tällöin avautuu ikkuna, jossa kysytään kentälle arvo, jonka jälkeen kenttä lisätään tekstiin. Kuvassa 13 on esitetty esimerkki täytetystä syöttökenttiä käyttävästä lomakkeesta.



Kuva 13: Esimerkki syöttökentillä toteutetusta lomakkeesta.

Tässä yksinkertaisessa esimerkissä kerätään OpenDocument-muotoisesta tekstiasiakirjasta kaikkien syöttökenttien sisältämät tiedot ja tulostetaan ne HTML-muotoisena taulukkona. Tulostusmuoto voisi olla yhtä helposti vaikkapa pilkuin erotettu taulukkotiedosto eli CSV-tiedosto. Alla esitetty ohjelma kerää syöttökentät mistä tahansa tiedostosta, ei vain lomakkeesta. Lomakkeen syöttökentät on tässä esimerkissä nimetty tunnisteina `k_sukunimi`, `k_etunimet` ja niin edelleen, joskin selkokielisiäkin tunnisteita voi käyttää, mikäli vastaanottava tietojärjestelmä hyväksyy sellaisia.

Muunnoksen XSLT-ohjelma on seuraavanlainen. Alussa on nimialueiden määrittelyjen vakio-osa, jonka jälkeen tulee tulosmuodon määrittely (HTML) ja lopulta kolme muunnossääntöä.

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!-- ============================================================= -->
<!-- OpenDocument-formaatin nimiavaruusmäärittelyt                 -->
<!-- ============================================================= -->
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:office="urn:oasis:names:tc:opendocument:xmlns:office:1.0"
  xmlns:text="urn:oasis:names:tc:opendocument:xmlns:text:1.0"
  office:version="1.0"
  version="1.0">

  <!-- Haluamme tulostaa HTML-muotoa. -->
  <xsl:output method="html" encoding="ISO-8859-1"/>

  <!-- ============================================================= -->
  <!-- Lue syöttökentän tiedot ja muotoile ne taulukon riviksi       -->
  <!-- ============================================================= -->
   <xsl:template match="text:text-input">
     <xsl:if test="normalize-space(.) != ''">
       <tr>
         <td><xsl:value-of
        select="normalize-space(@text:description)"/>:</td>
```

```
        <td><xsl:value-of select="normalize-space(.)"/></td>
      </tr>
    </xsl:if>
  </xsl:template>

  <!-- ============================================================ -->
  <!-- Juurisolmu                                                   -->
  <!-- ============================================================ -->
  <xsl:template match="/">
    <body>
      <h2>Lomakeraportti</h2>
      <p>Generoitu täytetyn lomakkeen sisältävästä OASIS OpenDocument
      -tekstiasiakirjasta.</p>

      <table>
        <tr>
          <th align="left">Tekstikenttä</th>
          <th align="left">Täytetty arvo</th>
        </tr>
        <xsl:apply-templates/>
      </table>
    </body>
  </xsl:template>

  <!-- ============================================================ -->
  <!-- Karsitaan pois kaikki muut tekstit                           -->
  <!-- ============================================================ -->
  <xsl:template match="text()">
  </xsl:template>

</xsl:stylesheet>
```

XSLT-ohjelman voi asentaa OpenOffice.orgiin vientisuodattimeksi, kuten edellä kuvattu, tai käyttää ulkopuolisella muuntimella. Tuloksena saadaan seuraavanlainen HTML-asiakirjan runko:

```
<body ...nimiavaruusmäärittelyjä...>
<h2>Lomakeraportti</h2>
<p>Generoitu täytetyn lomakkeen sisältävästä OASIS OpenDocument
-tekstiasiakirjasta.</p>
<table>
<tr>
<th align="left">Tekstikenttä</th>
<th align="left">Täytetty arvo</th>
</tr>
<tr>
<td>k_sukunimi:</td>
<td>Suuri</td>
</tr>
<tr>
<td>k_etunimet:</td>
<td>Aleksanteri</td>
</tr>
<tr>
<td>k_ammatti:</td>
<td>valloittaja</td>
</tr>
<tr>
<td>k_syntymaaika:</td>
<td>20.7.356 eaa</td>
</tr>
</table>
</body>
```

HTML-tuloste näyttää web-selaimessa seuraavalta:

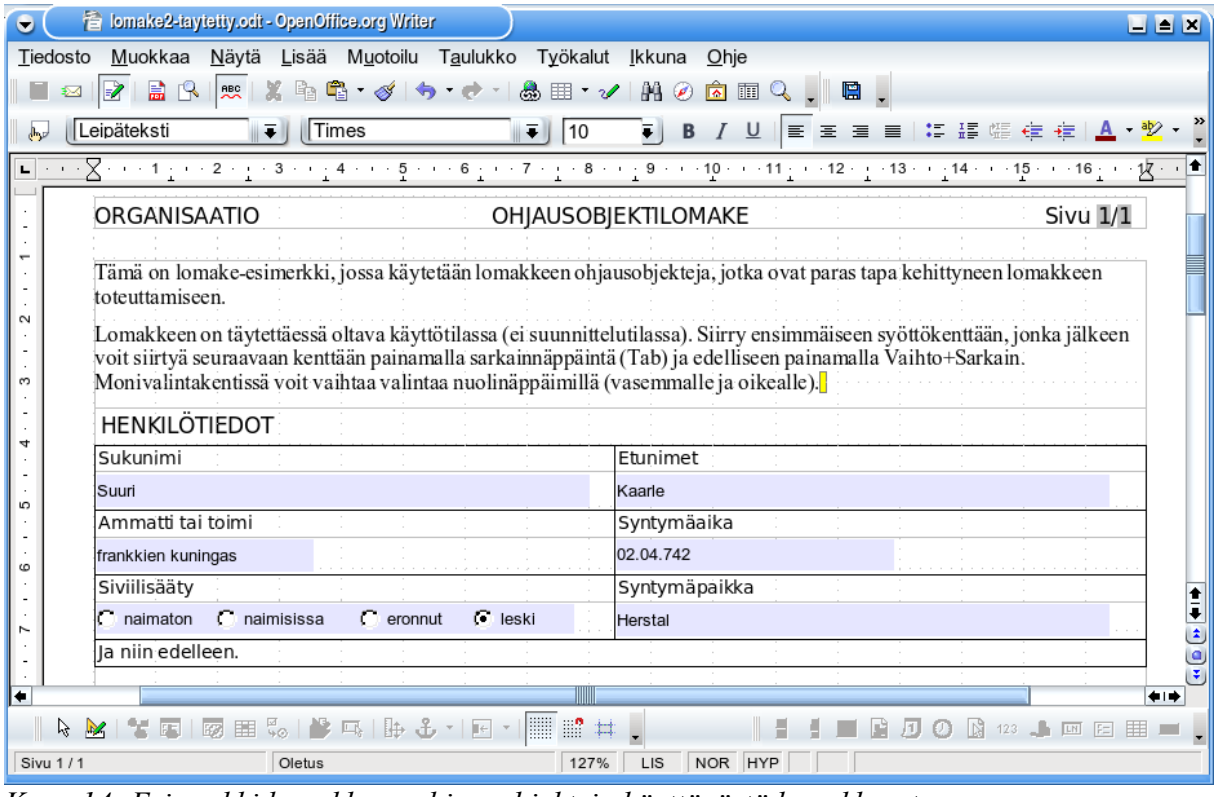
Lomakeraportti

Generoitu täytetyn lomakkeen sisältävästä OASIS OpenDocument -tekstiasiakirjasta.

Tekstikenttä	Täytetty arvo
k_sukunimi:	Suuri
k_etunimet:	Aleksanteri
k_ammatti:	valloittaja
k_syntymaaika:	20.7.356 eaa

8.6. Esimerkki 3: ohjausobjektilomakkeen tietojen kerääminen

Yllä käsittelimme tietojen keräämistä syöttökentillä tehdyistä lomakkeista. Syöttökentät ovat yleisyydestään huolimatta käyttäjän kannalta melko hankala tapa toteuttaa lomakkeita, sillä ne vaativat, että käyttäjä napsauttaa kutakin syöttökenttää erikseen. Syöttökenttien sisällöillä ei myöskään ole mitään rajoituksia esimerkiksi koon tai tyypin suhteen. Lomakkeen ohjausobjekteilla on mahdollista tehdä hyvin monipuolisia ja käyttöliittymälogiikaltaan hyvin hallittuja lomakkeita. Ohjausobjekteihin voidaan tehdä ohjauslogiikkaa makroilla tai XForms-lomakkeissa XPath-lausekkeilla. Lomake voidaan myös yhdistää tietokantaan, jolloin lomakkeeseen täytetyt tiedot kirjoitetaan tietokantaan suoraan (ks. luku 5.9, 4. käyttötapa). Tässä esimerkissä asiakirjan sisältämät tiedot kuitenkin kerätään muuntamalla ne toiseen tiedostomuotoon, tässä tapauksessa HTML:ksi.



Kuva 14: Esimerkki lomakkeen ohjausobjekteja käyttävästä lomakkeesta.

Oletetaan, että käyttäjä on täyttänyt lomakkeen kuvan 14 mukaisesti. Alla esitetty muunnos muuntaa lomakkeen HTML-muotoiseksi taulukoksi.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:office="urn:oasis:names:tc:opendocument:xmlns:office:1.0"
  xmlns:text="urn:oasis:names:tc:opendocument:xmlns:text:1.0"
  xmlns:form="urn:oasis:names:tc:opendocument:xmlns:form:1.0"
  office:version="1.0"
  version="1.0">

  <!-- Määrätään muunnoksen ulostulomuoto ja merkkikoodaus. -->
  <xsl:output method="html" encoding="ISO-8859-1"/>

  <!-- ============================================================ -->
  <!-- Luetaan kenttien tiedot ja muotoillaan ne taulukon riviksi  -->
  <!-- ============================================================ -->

  <!-- Tekstikenttä -->
  <xsl:template match="form:text">
    <tr>
      <td><xsl:value-of select="@form:name"/>:</td>
      <td><xsl:value-of select="@form:current-value"/></td>
    </tr>
  </xsl:template>

  <!-- Muotoiltu tekstikenttä -->
  <xsl:template match="form:formatted-text">
    <tr>
      <td><xsl:value-of select="@form:name"/>:</td>
      <td><xsl:value-of select="@form:current-value"/>
      (numeerinen päivämäärä 1.1.1900 laskien)</td>
    </tr>
  </xsl:template>

  <!-- Monivalintakenttä -->
  <xsl:template match="form:radio">
    <xsl:if test="@form:current-selected = 'true'">
      <tr>
        <td><xsl:value-of select="@form:name"/>:</td>
        <td><xsl:value-of select="@form:label"/></td>
      </tr>
    </xsl:if>
  </xsl:template>


  <!-- ============================================================ -->
  <!-- Juurisolmu                                                   -->
  <!-- ============================================================ -->
   <xsl:template match="/">
     <body>
       <h2>Lomakeraportti</h2>

       <p>Generoitu täytetyn lomakkeen sisältävästä OASIS OpenDocument
       -tekstiasiakirjasta.</p>

       <table>
         <tr>
           <th align="left">Tekstikenttä</th>
           <th align="left">Täytetty arvo</th>
         </tr>
         <xsl:apply-templates/>
       </table>
     </body>
   </xsl:template>
```

```
  <!-- ============================================================ -->
  <!-- Karsitaan pois kaikki muut tekstit                           -->
  <!-- ============================================================ -->
   <xsl:template match="text()">
   </xsl:template>

</xsl:stylesheet>
```

Ohjelman suorittaminen tapahtuu kuten edellä kuvattiin, joko OpenOfficeen asennetulla suotimella tai ulkoisesti. Suorittaminen tuottaa seuraavanlaisen raportin:

Lomakeraportti

Generoitu täytetyn lomakkeen sisältävästä OASIS OpenDocument -tekstiasiakirjasta.

Tekstikenttä	**Täytetty arvo**
k_ammatti:	frankkien kuningas
k_syntymaaika:	15438 (numeerinen päivämäärä 1.1.1900 laskien)
k_sukunimi:	Suuri
k_etunimet:	Kaarle
k_siviilisaaty:	leski
k_syntymapaikka:	Herstal

9. TUONTISUOTIMET

OpenOffice-ohjelmistoon on mahdollista rakentaa XML-muotoisten tiedostojen tuontisuotimia XSLT-kielellä. XSLT-kielestä ja siihen liittyvistä työkaluista kerrotaan tarkemmin liitteessä B. Tuontisuotimet voidaan asentaa OpenOfficeen tai niitä voidaan käyttää ulkoisissa ohjelmissa.

Tuonnin voi tehdä kahdella tavalla. XSLT-muunnin voi tulostaa joko pelkän `content.xml`-tiedoston tai koko OpenDocument-asiakirjan yhden XML-tiedoston muodossa. Mikäli se tulostaa vain `content.xml`-tiedoston, tarvitaan *tuontimalli*, joka sisältää asiakirjan muut tiedostot. Tämä tapa on yleensä helpompi, koska muut tiedostot ovat yleensä vakiosisältöisiä. Tällöin asiakirja voi sisältää helpommin myös kuvia.

Tuontisuotimen asennus on kuvattu edellä luvussa 7.1. Kun suodin on asennettu, se näkyy tiedoston avausikkunan **Tiedoston tyyppi** -valintalistassa. Mikäli suotimen määrityksissä on määritetty sen sen tukeman tiedostomuodon DocType-tunniste, tunnistetaan tiedosto automaattisesti suotimella käsiteltäväksi, mikäli DocType-merkkijono esiintyy sen alussa. Toissijaisesti tunnistuksessa käytetään tiedoston tarkennetta, jonka tulee olla suotimen asetusten mukainen.

9.1. Tuontisuodin ulkoisessa järjestelmässä

Tuontisuodinten käyttö ulkoisessa sovelluksessa tapahtuu paljolti kuten vientisuodintenkin (ks. luku 8.3), mutta päinvastaisesti. Ulkoisen sovelluksen täytyy luoda ZIP-paketti, joka sisältää tarvittavat tiedostot. Helpoiten tämä tapahtuu kuten OpenOffice-suotimissa, käyttäen tuontimallia, jonka pohjalta tuotu asiakirja luodaan korvaamalla sen `content.xml`-tiedosto XSLT-suotimen tulosteella.

Seuraavassa esimerkissä esitetään, kuinka XSLT-tuontisuodinta käytetään Linux-komentoriviltä käyttäen työkaluina Info-ZIP-pakkausohjelmaa (`zip`) ja Xalan-XSLT-suoritinta (ks. liite B.2). Käytämme suotimena alla esitettävää henkilötietokannan tuontisuodinta. Ensin kopioimme tuontimallin tekstiasiakirjaksi, jonka jälkeen teemme muunnoksen Xalan-suorittimen avulla ja korvaamme muunnoksen tuloksella asiakirjan ZIP-tiedoston sisältämän `content.xml`-tiedoston.

```
§ cp henkilotiedot-malli.ott henkilotiedot-testi.odt
$ xalan -in henkilotiedot.xml -xsl henkilotiedot-tuonti.xsl |
  zip -p henkilotiedot-testi.odt content.xml
updating: content.xml (deflated 73%)
```

9.2. Esimerkki: henkilötietokanta

Käytämme esimerkkinä hyvin yksinkertaista henkilötietokantaa, joka voidaan esittää seuraavanlaisena yksinkertaisena taulukkona:

Etunimi	**Sukunimi**	**Syntymäaika**	**Ammatti**	**Siviilisääty**
Aleksanteri	Suuri	20.7.356 eaa	valloittaja	naimaton
Kaarle	Suuri	2.4.742	kuningas	leski
Konstantinus	Suuri	27.2.273	keisari	naimisissa
Kyyrös II	Suuri	576 eaa	kuningas	eronnut

Esitämme tietokannan XML-muodossa seuraavasti. Henkilötietojen kentät ovat `<henkilo>`-elementin alaelementtejä, paitsi siviilisääty, joka esitetään attribuuttina.

```
<?xml version="1.0" encoding="UTF-8"?>
<henkilot>
  <henkilo>
    <etunimi>Aleksanteri</etunimi>
    <sukunimi>Suuri</sukunimi>
    <syntymaaika>20.7.356 eaa</syntymaaika>
    <ammatti>valloittaja</ammatti>
    <siviilisaaty tyyppi="naimaton"/>
  </henkilo>

  <henkilo>
    <etunimi>Kaarle</etunimi>
    <sukunimi>Suuri</sukunimi>
    <syntymaaika>2.4.742</syntymaaika>
    <ammatti>kuningas</ammatti>
    <siviilisaaty tyyppi="leski"/>
  </henkilo>

  <henkilo>
    <etunimi>Konstantinus</etunimi>
    <sukunimi>Suuri</sukunimi>
    <syntymaaika>27.2.273</syntymaaika>
    <ammatti>keisari</ammatti>
    <siviilisaaty tyyppi="naimisissa"/>
  </henkilo>

  <henkilo>
    <etunimi>Kyyrös II</etunimi>
    <sukunimi>Suuri</sukunimi>
    <syntymaaika>576 eaa</syntymaaika>
    <ammatti>kuningas</ammatti>
    <siviilisaaty tyyppi="eronnut"/>
  </henkilo>
</henkilot>
```

Esitämme seuraavaksi XSLT-pohjaisen tuontisuotimen, jonka avulla yllä esitetty XML-pohjainen henkilötietokanta muunnetaan OpenDocument-tekstiasiakirjaksi. Suodin koostuu seuraavista pääosista:

- asiakirjan rungon määrittelystä ja
- eri tyyppisten elementtien muunnoksista.

Tuontisuodin sisältää sekä XSLT-kielen elementtejä että OpenDocument-asiakirjan elementtejä. Nämä eri elementit erotetaan niiden nimiavaruuden pohjalta. XSLT-elementit on esitetty normaalilla fontilla ja muunnoksen tuloselementit kursiivilla.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:office="urn:oasis:names:tc:opendocument:xmlns:office:1.0"
  xmlns:text="urn:oasis:names:tc:opendocument:xmlns:text:1.0"
  office:version="1.0"
  version="1.0">

  <xsl:output indent="yes" encoding="UTF-8"/>

  <!-- ================================================================ -->
  <!-- Asiakirjan runko.                                                -->
  <!-- ================================================================ -->
  <xsl:template match="/">
    <office:document>
```

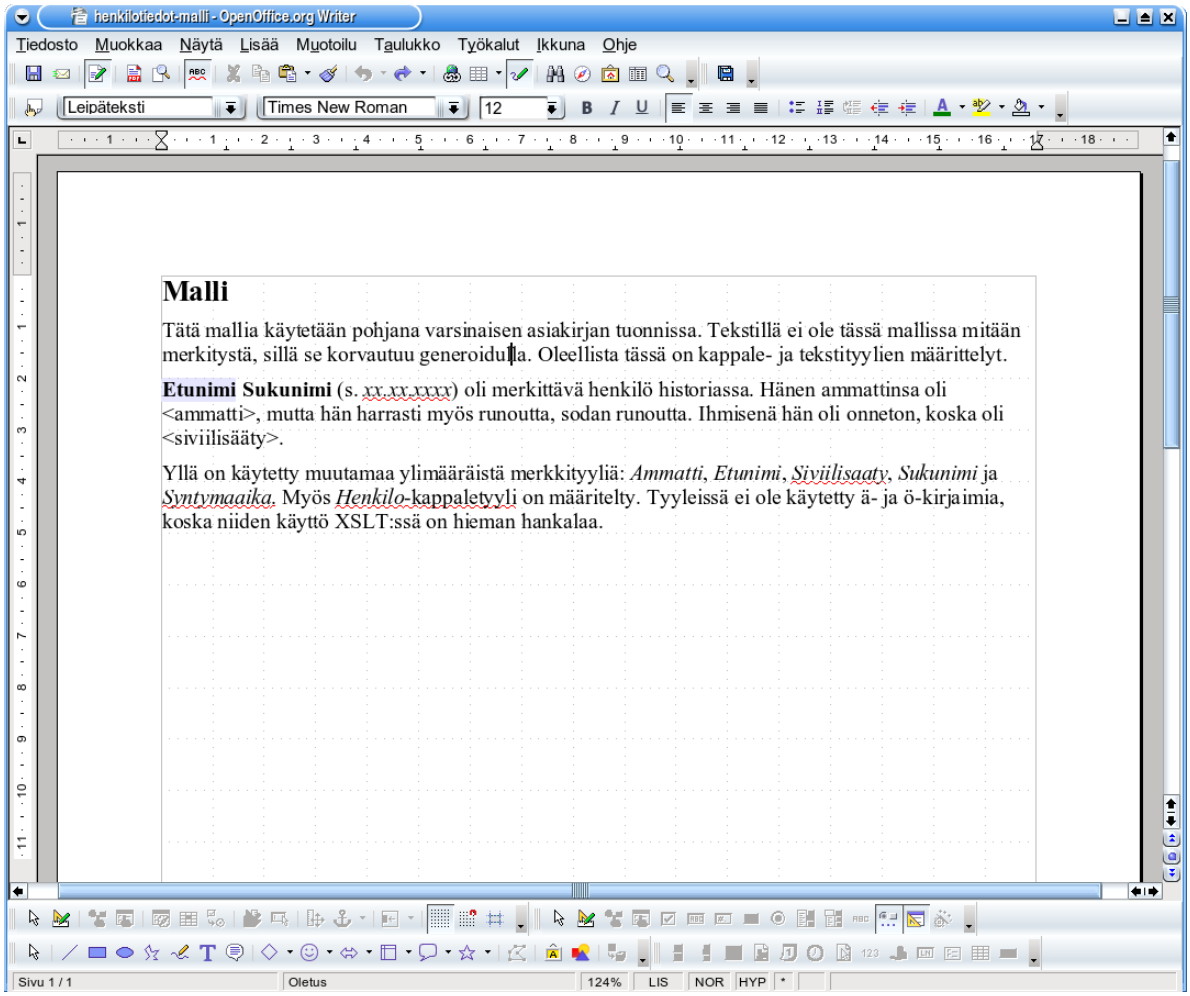
```
    <office:body>
      <office:text>

        <!-- Otsikko -->
        <text:h text:level="1" text:style-name="Heading 1">
          <xsl:text>Tuodut henkilötiedot</xsl:text>
        </text:h>

        <!-- Esittelyteksti -->
        <text:p text:style-name="Text body">
          <xsl:text>Tämä on omassa XML-formaatissa olevasta
          tiedostosta tuotu Writer-asiakirja.</xsl:text>
        </text:p>

        <!-- Käsitellään henkilöt -->
        <xsl:apply-templates select="henkilot/henkilo"/>

        <!-- Selityskappale -->
        <text:p text:style-name="Text body">
          <xsl:text>XML-tiedostosta ladatut tietueet ovat
          Henkilo-tyylisinä kappaleina. Tietueen kentät käyttävät
          nimettyjä merkkityylejä, joiden perusteella ne tunnistetaan
          tallennettaessa.</xsl:text>
        </text:p>

      </office:text>
    </office:body>
  </office:document>
</xsl:template>

<!-- ======================================================================= -->
<!-- Henkilön tiedot                                                         -->
<!-- ======================================================================= -->
<xsl:template match="henkilo">
  <text:p text:style-name="Henkilo">

    <!-- Etunimi -->
    <text:span text:style-name="Etunimi">
      <xsl:value-of select="etunimi"/>
    </text:span>
    <xsl:text> </xsl:text>

    <!-- Sukunimi -->
    <text:span text:style-name="Sukunimi">
      <xsl:value-of select="sukunimi"/>
    </text:span>

    <!-- Syntymäaika -->
    <xsl:text> (s. </xsl:text>
    <text:span text:style-name="Syntymaaika">
      <xsl:value-of select="syntymaaika"/>
    </text:span>
    <xsl:text>)</xsl:text>

    <!-- Muuta tekstiä -->
    <xsl:text> oli merkittävä henkilö historiassa. Hän oli </xsl:text>

    <!-- Ammatti -->
    <text:span text:style-name="Ammatti">
      <xsl:value-of select="ammatti"/>
    </text:span>

    <!-- Muuta tekstiä -->
    <xsl:text>, mutta harrasti myös runoutta, sodan runoutta. </xsl:text>
    <xsl:text>Ihmisenä hän oli onneton, koska oli </xsl:text>

    <!-- Siviilisääty -->
    <text:span text:style-name="Siviilisaaty">
```

```
        <xsl:value-of select="siviilisaaty/@tyyppi"/>
      </text:span>

      <!-- Piste kappaleen loppuun. -->
      <xsl:text>.</xsl:text>
    </text:p>
  </xsl:template>

</xsl:stylesheet>
```

Esimerkissä muotoillaan XML-syötteen `henkilo`-tietueet tekstikappaleiksi OpenDocument-asiakirjassa. Tekstiasiakirjojen kappaleet kirjoitetaan `<text:p>`-elementtien sisälle, kuten on kuvattu edellä luvussa 6.9. Attribuutti `text:style-name` määrittelee kappaleen tyylin, jonka on oltava joko `content.xml`-tiedoston alussa määritelty automaattinen tai tuontimallissa (ks. alla) määritelty varsinainen tyyli. XML-asiakirjan `henkilo`-elementti muunnetaan `Henkilo`-kappaletyylin mukaiseksi kappaleeksi.

Tietueiden kentät esitetään tietyllä merkkityylillä `<text:span>`-elementtien sisällä olevina tekstijaksoina. Tekstijakson merkkityyli määritetään `text:style-name`-attribuutilla. Merkkityylit on määritelty tuontimallin `styles.xml`-tiedostossa.

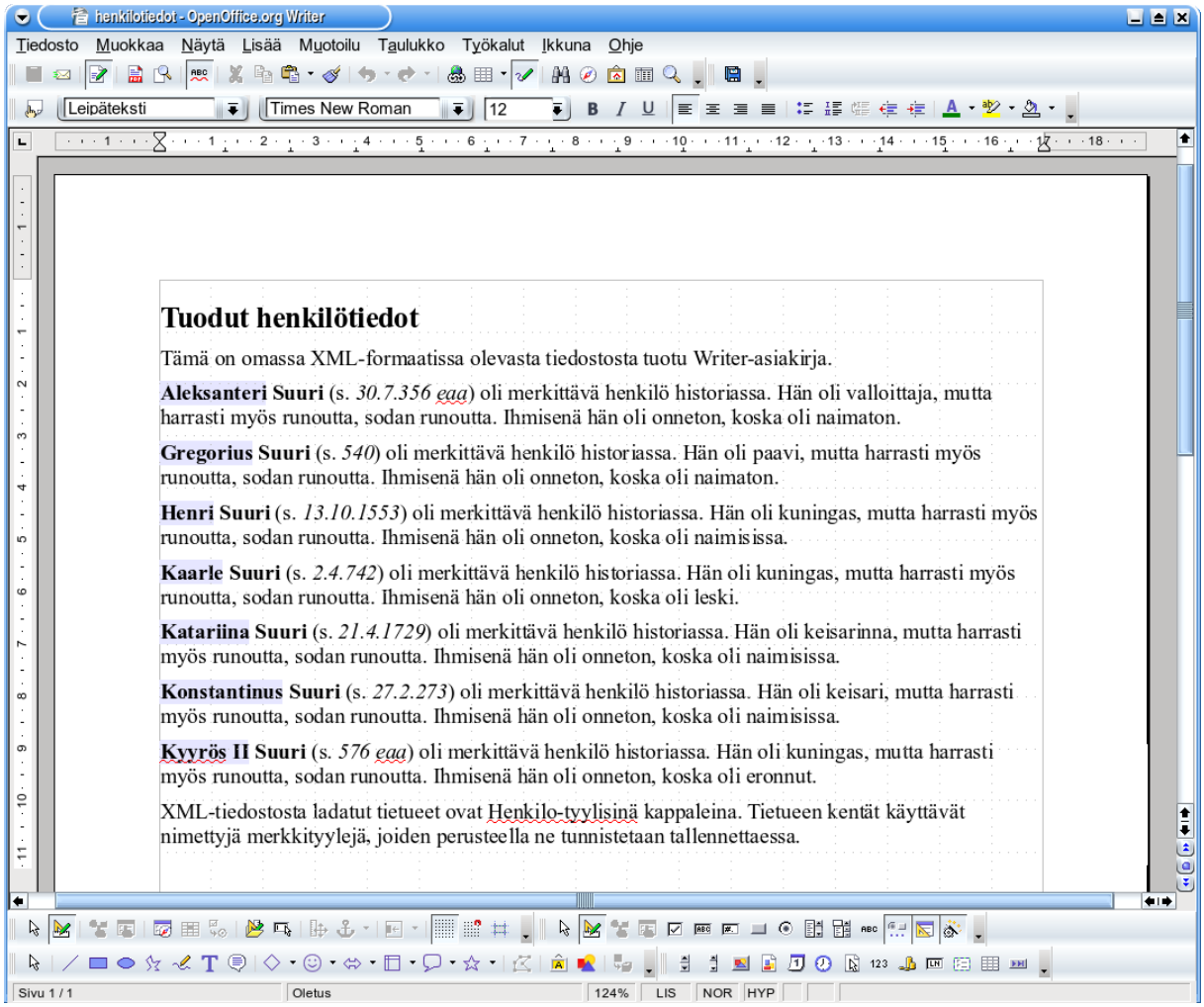
Tuontisuodin luo vain asiakirjan varsinaisen sisällön sisältävän `content.xml`-tiedoston. Kaikki muu sisältö, kuten epäautomaattiset (eli varsinaiset) tyylit sisältävä `styles.xml`-tiedosto, saadaan erillisestä tuonnin malliasiakirjasta. Tosin epäautomaattiset tyylit on luvallista määritellä myös `content.xml`-tiedoston alussa.



Kuva 15: Tuontimalli, joka sisältää kappale- ja merkkityylien määritykset.

Malliasiakirjan `content.xml`-tiedoston sisällöllä ei ole mitään merkitystä ja se korvataan tuotaessa. Sitä voi kuitenkin käyttää tyylejä suunniteltaessa esimerkkinä, kuten kuvassa 15 on tehty. XML-asiakirjaa muunnettaessa kannattaa yleensä määritellä kullekin tietue- ja kenttätyypille oma kappale-, sivu- tai merkkityyli, riippuen XML-asiakirjan rakenteesta. Henkilötietoesimerkissämme XML-asiakirjan `henkilo`-elementti muunnetaan `Henkilo`-kappaletyyliksi ja sen alaelementit merkkityyleiksi.

Tuonti voidaan tehdä OpenOffice.orgin sisällä asentamalla XSLT-muunnin suodattimeksi tai ulkopuolella erillisellä XSLT-käsittelijällä, kuten edellä on kuvattu. Yllä esitetyn tuontisuotimen avulla tuotu asiakirja on esitetty alla kuvassa 16.



Kuva 16: Henkilötietoesimerkin tuontisuotimen ja esimerkkitietokannan tuloksena oleva tekstiasiakirja.

10. OPENDOCUMENT MUILLA OHJELMOINTIKIELILLÄ

Ulkoisilla ohjelmilla on mahdollista käsitellä OpenDocument-asiakirjoja kahdella tavalla, joko käsittelemällä suoraan tiedostoja tai toimisto-ohjelmiston, kuten OpenOfficen, tarjoaman rajapinnan kautta.

Koska OpenDocument on XML-pohjainen tiedostomuoto, voidaan käsittelyyn käyttää tavanomaisia XML-rajapintoja, kuten SAX (Simple API for XML) ja DOM (Document Object Model). SAX-rajapinta perustuu elementtien ja niiden tietosisällön lukemiseen yksisuuntaisesti tiedoston alusta loppuun. SAX tarjoaa ne käsittelyohjelmalle yksitellen callback-funktiolla. DOM-rajapintaa käytettäessä koko asiakirja taas luetaan kerralla muistiin ja sen elementteihin voidaan osoittaa vapaasti erilaisilla osoitustavoilla.

Koska OpenDocument-asiakirja on ZIP-pakattu, täytyy sen käsittelyyn käyttää ZIP-purku- ja pakkausohjelmaa tai ZIP-ohjelmointirajapintaa.

OpenDocument-asiakirjojen käsittelystä eri ohjelmointikielillä (XSLT, Java ja Perl) kerrotaan laajasti kirjassa OASIS OpenDocument Essentials [Eis2006]. Kirjasta ja siihen liittyvästä verkosta ladattavasta oheismateriaalista löytyy lukuisia eri kielillä tehtyjä esimerkkejä.

10.1. Python

Python on eräs OpenOfficen kannalta tärkeimmistä kielistä. Se asennetaan OpenOffice-asennuksen mukana ja mukana tulee myös Python-UNO-silta, joka mahdollistaa OpenOfficen hallinnan Python-ohjelmista käsin. UNO-rajanpinnan käytöstä Python-kielellä kerrotaan tarkemmin luvussa 11.

Pythonin peruskirjastot tarjoavat erinomaiset työkalut OpenDocument-asiakirjojen kannalta keskeisten ZIP- ja XML-tiedostojen käsittelyyn. ZIP-tiedostoa voi käsitellä Pythonin `zipfile`-moduulilla. XML-tiedostoja voi käsitellä `xml.sax`-moduulilla tai tiedoston voi lukea DOM-dokumentiksi `xml.dom`-moduulilla.

Seuraava SAX-rajapintaa käyttävä ohjelma tulostaa asiakirjan otsikot jäsennystason mukaan sisennettynä:

```
import sys, zipfile, xml.sax

# SAX-käsittelijäluokka
class SaxKasittelija (xml.sax.ContentHandler):
    def __init__ (self):
        self.mTaso     = 0   # Otsikon jäsennystaso
        self.mTeksti   = ""  # Otsikon tekstisisältö

    # Elementin alku
    def startElement (self, name, attrs):
        # Jos elementti on otsikkoelementti text:h
        if name == "text:h":
            # Tallennetaan attribuutissa annettu jäsennystaso
            self.mTaso = int (attrs.getValue ("text:outline-level"))
            self.mTeksti = ""

    # Luetaan nykyisen elementin merkkisisältöä muuttujaan
    def characters (self, content):
        self.mTeksti += content

    # Elementin loppu
    def endElement (self, name):
```

```
        # Jos päättyvä elementti on otsikkoelementti
        if name == "text:h":
            # Tulostetaan kerätty otsikon tekstisisältö,
            # sisennettynä jäsennystason mukaan
            print (" "*(self.mTaso-1)) + self.mTeksti
            self.mTaso   = 0
            self.mTeksti = ""

########################################################################
# Pääohjelma
########################################################################

# Avataan OpenDocument-tiedosto
odt = zipfile.ZipFile (sys.argv[1], "r")

# Luetaan lomakeasiakirjan content.xml-tiedosto merkkijonoksi
sisalto = odt.read ("content.xml")

# Luodaan käsittelijäobjekti
saxhandler = SaxKasittelija ()

# Käsitellään XML-asiakirja
xml.sax.parseString (sisalto, saxhandler)
```

Ohjelma voidaan suorittaa Linux- tai UNIX-käyttöjärjestelmissä esimerkiksi seuraavanlaisella komennolla:

```
$ python otsikot.py OM_ODF_raportti.odt
JOHDANTO
AVOIN TIEDOSTOMUOTO
OPENDOCUMENT-STANDARDIN KEHITTYMINEN
 OpenDocument-standardin käyttöönottotietoja
EU-SUOSITUKSET AVOIMISTA TIEDOSTOMUODOISTA
...
```

10.2. Komentorivityökalut

Koska XML-tiedostot ovat tekstipohjaisia, on niiden käsittely helppoa useimmilla tekstin prosessointiin soveltuvilla ohjelmointikielillä ilman varsinaista XML:n jäsentämistäkin. Tämä tapa sopii moniin yksinkertaisiin käsittelytehtäviin, kuten tietyn tyyppisten elementtien listaamiseen tai lyhyiden tekstinosien tai XML-attribuuttien korvaamiseen. Tyypillinen XML-attribuuttien korvaus on linkitettyjen kuvatiedostojen polkujen muuttaminen, mihin tehtävään XSLT-suodin voi olla turhan työläs rakentaa.

Tarkastelemme siksi OpenDocument-asiakirjojen käsittelyä yksinkertaisilla Linux- ja UNIX-käyttöjärjestelmien komentorivityökaluilla. Koska tavanomaiset UNIX-komentorivityökalut soveltuvat parhaiten tekstitiedostojen käsittelyyn riveittäin, täytyy OpenOfficen asetuksista **Työkalut > Asetukset > Lataus ja tallennus > Yleistä** olla asetus **Koon optimointi XML-muodolle** poissa päältä.

OpenDocument-paketin avaaminen tapahtuu Info-ZIP-ohjelmalla (`unzip`-komennolla). Jos halutaan purkaa vain yksi tiedosto, tyypillisesti `content.xml`, annetaan se asiakirjatiedoston jälkeen. Valitsin `-p` aiheuttaa puretun sisällön tulostamisen oletustulosteeseen, jolloin se voidaan putkittaa käsittelyohjelmille, kuten `grep` ja `sed`.

```
$ unzip -p OM_ODF_raportti.odt content.xml | grep '\<text:h\>'
```

Seuraava pidempi komento purkaa (`unzip`) odt-tiedoston, seuloo siitä ensimmäisen tason otsikot (`grep`), muuntaa UTF-8 merkistön ISO-8859-15-merkistöön (`iconv`) ja poistaa XML-

merkinnät ja ylimääräiset välilyönnit alusta (`sed`). Lopputuloksena on siisti lista lukujen otsikoita.

```
$ unzip -p OM_ODF_raportti.odt content.xml | grep '\<text:h\>' |grep
'level="1"'| iconv -f UTF-8 -t ISO-8859-15|sed -e 's/<[^>]\+>//g'|sed -e
's/^ \+//'
JOHDANTO
AVOIN TIEDOSTOMUOTO
OPENDOCUMENT-STANDARDIN KEHITTYMINEN
EU-SUOSITUKSET AVOIMISTA TIEDOSTOMUODOISTA
...
```

11. OPENOFFICEN ULKOINEN HALLINTA

Ulkoiset ohjelmat voivat hallita OpenOfficea UNO-rajapinnan kautta. Ulkoiset ohjelmat voivat luoda OpenDocument-asiakirjoja, muokata niitä, tallentaa niitä eri muotoihin ja lukea asiakirjojen sisältöä. OpenOfficea voidaan tällöin käyttää myös ilman käyttöliittymää palvelintilassa esimerkiksi asiakirjamuunnosten tekemiseen.

11.1. UNO-komponenttimalli

UNO (Universal Network Objects) on OpenOfficen komponenttimalli, joka mahdollistaa yhteentoimivuuden eri ohjelmointikielten, laitearkkitehtuurien ja toisten komponenttimallien välillä. UNOa voi käyttää prosessien sisällä, mutta se toimii myös hajautetussa arkkitehtuurissa sekä prosessien välillä että niin intra- kuin Internetissäkin. UNO-rajapinnalle on sidokset ainakin seuraaville kielille ja komponenttimalleille:

C++	OpenOffice on kehitetty C++-kielellä, joka on myös tehokkain kieli komponenttien toteuttamiseen. Esimerkiksi suomen kielen oikolukukomponentit Soikko ja Voikko on kehitetty C++-kielellä. Vaikeutena on, että komponentit täytyy kääntää jokaiselle laitearkkitehtuurille ja käyttöjärjestelmälle erikseen.
StarBasic	StarBasic on OpenOfficen sisäänrakennettu makrokieli. OpenOfficessa on sisäänrakennettuna kehitys- ja virheenjäljitystyökalut Basic-kielelle.
Java	Monet OpenOfficen komponentit, kuten Base ja monet ohjatut toiminnot, on kehitetty Javalla. Javan käyttö vaatii Java-ajoympäristön asennuksen.
Python	Tarjoaa yhteentoimivimman ja laitearkkitehtuurista ja käyttöjärjestelmästä riippumattoman ulkoisen hallintakeinon. Python/UNO-silta ja ajoympäristö tulevat OpenOfficen mukana, joten Python on käytettävissä kaikissa asennuksissa.
OLE	Mahdollistaa UNO-komponenttien käytön COM-komponenttimallin kautta Windows-käyttöjärjestelmässä.
CLI	Mahdollistaa UNO-komponenttien käytön Microsoft .NET -ympäristöstä.

11.2. OpenOfficen etähallinta

Jotta ulkoinen ohjelma voisi hallita OpenOfficea, täytyy OpenOfficessa avata UNO-hyväksyjä, jonka kautta UNO-kutsuja voi tehdä. Tämä tapahtuu käynnistämällä `soffice`-ohjelma `-accept=<hyväksyjämerkkijono>`-parametrilla. Esimerkiksi Linux- tai UNIX-käyttöjärjestelmässä sen voi tehdä komentoriviltä seuraavasti:

```
$ soffice "-accept=socket,host=localhost,port=8100;urp;"
```

Mikäli OpenOffice on ennestään käynnissä, komento vain lisää käynnissä olevaan OpenOfficeen uuden hyväksyjän. Hyväksyjän merkkijono on puolipistein eroteltu lista parametreja: yhteys;protokolla;objektin nimi. Sekä yhteydellä että protokollalla voi olla parametrejä, jotka erotetaan pilkuin.

```
[yhteys],parametrit,...;[protokolla],parametrit,...;objektinimi
```

Yhteys-parametrin arvona on yleensä ”`socket`” eli pistoke, joka saa parametreikseen isäntänimen ja porttinumeron. Tavallisin tapaus on hyväksyä yhteydet samalta tietokoneelta,

jolloin isäntänimeksi määritetään `localhost`. Isäntänimen voi myös antaa IP-osoitteena. Mikäli arvo on 0.0.0.0 (tai vain 0), hyväksytään yhteydet mistä tahansa tietokoneesta, mikä saattaa myös aiheuttaa tietoturvariskin. Porttinumero on vapaavalintainen väliltä 1000-10000. Protokollana käytetään yleensä `urp`-protokolaa. Viimeisenä parametrina tulee objektin nimi, johon voidaan ottaa yhteys. Mikäli objektin nimeä ei anneta, voidaan yhteys ottaa mihin tahansa objektiin.

Monissa OpenOfficea ulkoa käyttävissä sovelluksissa, kuten asiakirjamuunnoksissa tai tulostuksessa, ei tarvita lainkaan käyttöliittymää. OpenOffice voidaan tällöin käynnistää palvelintilassa eli ilman käyttöliittymää `-headless` -komentoriviparametrilla.

11.3. Esimerkki: asiakirjan avaaminen

Esitämme alla Python-kielisen ohjelman, joka avaa asiakirjan ja kirjoittaa sen loppuun tekstiä. Asiakirja avataan käyttäjälle muokattavaksi (OpenOfficea ei ajeta palvelintilassa). Asiakirjan nimi annetaan ohjelmalle parametrina. Parametrina annettava tiedosto voi olla myös malli, jolloin sen pohjalta luodaan uusi asiakirja.

```
# Asetetaan Python-ohjelman merkkikoodaus
# -*- coding: iso-8859-15 -*-

# Käytettävät moduulit
import uno     # Käytetään UNO-siltaa
import sys     # Käytetään järjestelmäfunktioita
import os.path # Tarvitaan tiedostonimen muuttamiseen URL-osoitteeksi
from com.sun.star.beans import PropertyValue

###############################################################################
# Asetukset
###############################################################################

# OpenOffice-prosessin yhteysportti
sofficeport = 2009

# Luetaan komentoriviltä avattavan tiedoston nimi
# ja muunnetaan se absoluuttiseen muotoon
doc_filename = os.path.abspath(sys.argv[1])

###############################################################################
# Luodaan UNO-yhteys käynnissä olevaan OpenOffice-prosessiin
###############################################################################

# Noudetaan UNO-komponenttikonteksti PyUNO-sillalta
context = uno.getComponentContext()

# Luodaan UNO-hyväksyjä-objekti
resolver = context.ServiceManager.createInstanceWithContext
("com.sun.star.bridge.UnoUrlResolver", context)

# Yritetään ottaa yhteys käynnissä olevaan OpenOffice-prosessiin
try:
    ctx =
resolver.resolve("uno:socket,host=localhost,port=%d;urp;StarOffice.ComponentC
ontext" % (sofficeport))
except:
    # Yhteyden otto epäonnistui, tulostetaan ohjeteksti ja poistutaan
    print "OpenOffice ei vastaanota UNO-yhteyksiä portissa %d localhost-
osoitteessa." % (sofficeport)
    print "Käynnistä OpenOffice-prosessi seuraavalla komennolla: "
    print "    soffice \"-accept=socket
,host=localhost,port=%d;urp;StarOffice.ServiceManager\"" % (sofficeport)
    sys.exit(1)
```

```
# Pyydetään palvelunhallinta-objekti
servicemanager = ctx.ServiceManager

# Luodaan desktop-instanssi
desktop =
servicemanager.createInstanceWithContext("com.sun.star.frame.Desktop",ctx)

################################################################################
# Avataan asiakirja
################################################################################

# Muunnetaan tiedoston nimi UNO-rajapinnan vaatimaan URL-muotoon
doc_url  = uno.systemPathToFileUrl (doc_filename)

# Määritellään latauksen parametrit
# - Hidden-asetus arvoon False määrää, että asiakirjaa ei avata piilotettuna
properties = tuple([PropertyValue("Hidden", 0, False, 0)])

# Ladataan asiakirja tiedostosta
doc = desktop.loadComponentFromURL(doc_url, "_blank", 0, properties)

################################################################################
# Käsitellään asiakirjaa
################################################################################

# Otetaan asiakirjan tekstiobjekti
text = doc.Text

# Luodaan kohdistin
cursor = text.createTextCursor()

# Siirretään kohdistin asiakirjan loppuun
cursor.gotoEnd(False)

# Lisätään asiakirjaan tekstiä osoittimen kohdalle
text.insertString(cursor, "Tässä on tekstiä", 0)

# Lopuksi pakotetaan UNO-kutsujen välimuistissa olevien kutsujen suoritus
# ennen Python-ohjelmasta poistumista. (Ks. Python-UNO Tutorial)
ctx.ServiceManager
```

Ohjelman suoritus edellyttää, että OpenOffice-prosessi on asetettu vastaanottamaan komentoja portissa 2009 (tai muulla ohjelmassa käytetyllä). Tämä vaatii `soffice`-ohjelman käynnistämistä alla esitetyillä parametreilla. Käynnistyksen voi tehdä UNO-rajapintaa käyttävästä ohjelmasta tai monella muulla tavalla, mutta esitämme tässä miten se tehdään Linux- ja UNIX-käyttöjärjestelmissä komentoriviltä. Annetaan seuraava komento:

```
$ soffice "-accept=socket,host=localhost,port=2009;urp;"
```

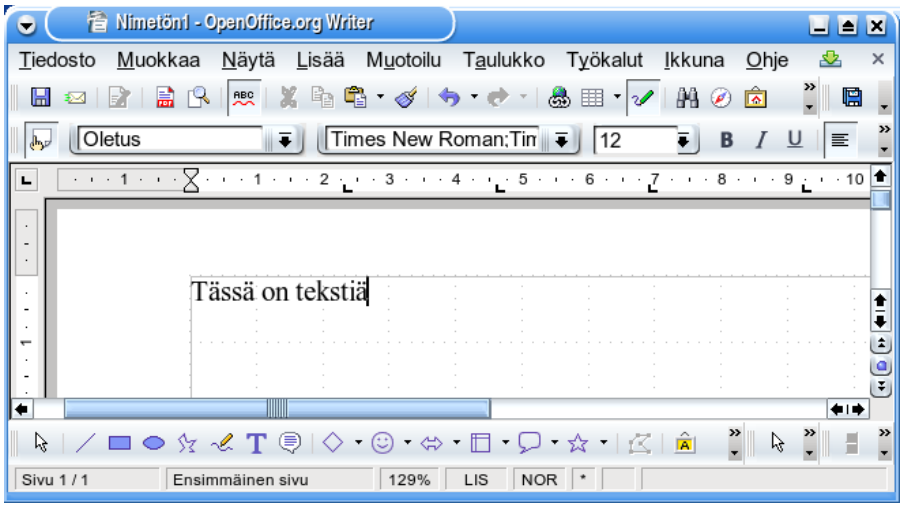
Mikäli OpenOffice-prosessi on ennestään käynnissä, komento vain lisää siihen parametrien mukaisen käsittelijän ja poistuu. Muutoin se käynnistää OpenOffice-prosessin ja komennon suoritus päättyy vasta, kun OpenOffice-prosessi lopetetaan.

Python-ohjelman suorittaminen vaatii PyUNO-sillan ja muut tarpeelliset kirjastot. Näitä ei ole normaalin Python-version asennuksessa mukana, joten helpointa on käyttää OpenOfficen mukana asennettavaa Python-tulkkia. Python-tulkki käynnistetään OpenOfficen asennushakemiston alla olevasta `program`-hakemistosta. Ohjelma voidaan käynnistää monella tavoin, mutta esitämme tässä, miten se käynnistetään käytettäessä komentoriviä Linux- tai UNIX-käyttöjärjestelmässä. Tämä tapa ei sovi tavalliselle käyttäjälle, joten käytännön tapauksessa ohjelma täytyisi asentaa käytettävään käyttöjärjestelmään sopivalla tavalla.

Oletetaan, että yllä esitetty ohjelma on nimetty `openodf.py` ja nykyisessä kansiossa on mallipohja nimeltä `malli.ott`. Ohjelma suoritetaan tällöin seuraavasti:

```
§ /opt/openoffice.org2.3/program/python openodf.py malli.ott
```

Ohjelman luoma ja käyttäjälle avaama asiakirja on esitetty kuvassa 17 alla.



Kuva 17: Ulkoisen ohjelman mallipohjasta luoma uusi asiakirja, johon ohjelma on kirjoittanut samalla tekstiä.

12. XFORMS-LOMAKKEET

XForms on XML-pohjainen merkintäkieli, joka on tarkoitettu tietolomakkeiden kenttien määrittelyyn ja kenttien yhdistämiseen XML-tietueisiin. XFormsin tärkein käyttökohde ovat lomakkeet, joiden sisältö on mahdollista lähettää sähköisesti.

XForms on alun perin suunniteltu HTML-lomakkeiden uudeksi sukupolveksi, mutta se on riittävän yleiskäyttöinen soveltuakseen myös muunlaisiin ympäristöihin. Sen etuja HTML-lomakkeisiin nähden ovat mm.

- Vahva tyypitys – syöttöarvojen tarkistus ilman varsinaista ohjelmointia
- Tietojen lähetys palvelimelle
- Sisällön ja esitystavan eriyttäminen

XForms on WWW-standardeista vastaavan World Wide Web Consortium (W3C) -standardointielimen julkaisema standardi.

Tämä OpenDocument-tiedostomuodon käyttökohde on paljolti sama kuin PDF-lomakkeiden, jotka myös mahdollistavat älykkään lomakelogiikan ja tietojen lähettämisen XML-muodossa. PDF-lomakkeiden etuna on sen vaatiman Adobe Reader -ohjelman valtaisa yleisyys. XForms-lomakkeen sisältävän OpenDocument-asiakirjan voi myös tallentaa ja avata myöhemmin uudelleen – Adobe Reader -ohjelman perusversiossa tätä ominaisuutta ei ole, vaan vain maksullisessa versiossa. OpenDocument-pohjaista XForms-lomaketta on myös mahdollista muokata käyttäjän toimesta, vaikkakaan sen ei yleensä pitäisi olla tarpeellista tai edes toivottavaa.

Tietojen esitäyttäminen tapahtuu siten, että tietojärjestelmä täyttää tiedot OpenDocument-asiakirjaan tallennettuun XML-instanssiin, mutta täyttäminen voi myös tapahtua HTTP-kyselyllä. Täyttämisen jälkeen käyttäjä joko tulostaa asiakirjan tai painaa lähetyspainiketta, joka lähettää tiedot XML-muodossa kohdetietojärjestelmään.

XForms-lomakkeen esitäytetyt tiedot voidaan lukea suoraan verkosta tai XML-tiedostosta, eivätkä vaadi OpenDocument-asiakirjan generointia. Tosin avain, jonka perusteella tiedot haetaan, täytyy olla asiakirjassa.

XForms tai PDF-lomakkeet eivät kumpikaan ole paras ratkaisu kaikkiin lomaketarpeisiin, vaan ovat niche-välineitä, jotka toimivat rajatuissa käyttötarkoituksissa ja tietojärjestelmäympäristöissä. Useimmat tietokantajärjestelmät, jotka vaativat tietojen syöttämistä, tarjoavat omat lomakekäyttöliittymänsä. Esimerkiksi web-selainpohjaiset sovellukset käyttävät pääsääntöisesti HTML-lomakkeita, joiden kehittämiseen on nykyään saatavilla erittäin monipuolisia ja kehittyneitä työkaluja. Lomakeratkaisujen alustavaihtoehtoja käsitellään myös edellä luvussa 5.9.

OpenOffice tukee XForms-lomakkeita melko kattavasti. Oleellisin puuttuva ominaisuus on toistuvat elementit, joiden avulla voidaan tehdä esimerkiksi luetteloita. Tämän puutteen saa kierrettyä luomalla elementit manuaalisesti ”riittävälle” määrälle.

12.1. Malli-näkymä-ohjain-arkkitehtuuri

XForms noudattaa malli-näkymä-ohjain-arkkitehtuuria, joka eriyttää tiedon esityksen

käyttöliittymästä ja hallintalogiikasta. Malli käsittää tiedon esityksen: XFormsissä tiedon esitys pohjautuu XML:ään ja käsitellään XML-instanssina. OpenOfficen ohjausobjektit toimivat näkymänä. Ohjaus tapahtuu sidoksissa, jotka määrittelevät näkymän ja mallin väliset suhteet ja toimintalogiikan XPath-lausekkeina.

Yksinkertaisimmillaan XForms-lomakkeet toimivat kuten HTML-lomakkeet, joihin syötetään arvoja ja lähetetään palvelimelle. XForms-tietokenttien arvot voidaan kuitenkin tarkistaa välittömästi ilman JavaScript-ohjelmointia.

Instanssi on XML-rakenne, jossa olevia tietoja XForms-lomakkeessa voidaan käsitellä. Instanssien tiedot voivat olla tallennettuja OpenDocument-asiakirjaan tai OpenOffice voi lukea linkitetyn instanssin HTTP-kyselyn avulla palvelimelta.

Sidos on XPath-kielellä tehty määrittely, jossa viitataan instanssitietoihin. Sidoksen arvolla on tietotyyppi, kuten instanssin kentälläkin. Sidoksella voi olla lisäksi seuraavat ehdot:

Pakollinen	Jos tosi, ei arvoa saa jättää tyhjäksi.
Relevantti	Jos epätosi, tietoja ei tarvita ja sidokseen viittaavat ohjausobjektit näkyvät harmaina. Jos tosi, ohjausobjekti toimii normaalisti.
Rajoite	Lauseke, jonka täytyy olla tosi, jotta arvo olisi hyväksyttävä.
Kirjoitussuojattu	Jos tosi, arvoa ei voi muuttaa.
Laskenta	Suorittaa laskutoimituksen ja tallentaa arvon sidottuun kenttään.

Näkymä rakennetaan OpenDocument-tiedostomuodon tukemilla lomakkeen ohjausobjekteilla.

13. YHTEENVETO

Muokkauskelpoisten toimistoasiakirjojen tallennusmuodoksi kehitetyn OpenDocument-tiedostomuodon tultua hyväksytyksi ensin OASIS-standardiksi vuonna 2005 ja vuotta myöhemmin ISO-standardiksi ISO/IEC 26300 se on yleistynyt nopeasti organisaatioiden käytössä. Tiedostomuoto on lisäksi jo useiden merkittävien ohjelmistojen tukema monilla käyttöjärjestelmäalustoilla.

Useissa EU-maissa on tehty avointa OpenDocument-standardia tukevia kansallisia linjauksia. Suomen julkishallinnolta puuttuu toistaiseksi vielä kansallinen linjaus toimistoasiakirjojen tiedostomuodoista ja EU:n sähköisen hallinnon periaatteisiin kuuluvasta organisaatioiden välisestä yhteentoimivuuskehyksestä. Oikeusministeriön vuonna 2006 tekemä päätös siirtyä avoimen lähdekoodin OpenOffice.org-ohjelmistoon ja OpenDocument-tiedostomuodon käyttöön on toistaiseksi merkittävin suomalainen suuren organisaation OpenOffice- ja OpenDocument-käyttöönotto.

OpenDocument-raportti on laadittu oikeusministeriön OpenOffice- ja OpenDocument-käyttöönoton yhteydessä suomenkieliseksi perusinformaatioksi tiedostomuodosta ja sen hyväksikäytön tietoteknisistä perusteista. Raportissa on kuvattu yleisellä tasolla OpenDocument-tiedostomuoto, sen rakenne sekä kehittämisen ja standardoinnin tausta. Raportissa on myös yhteenveto tiedostomuodon käyttöönottotilanteesta ja avointen tiedostomuotojen EU-tasoisista suosituksista.

Raportin teknisessä osuudessa on kuvattu tarkemmin tiedostomuodon rakenne eri asiakirjatyypeille. Tiedostomuodon hyväksikäyttöä on kuvattu useilla käyttötapauksilla ja niiden toteutuksen periaatteilla. Käyttötapauksissa käsitellään asiakirjojen käsittelyn integrointia organisaation tietojärjestelmiin ja integroinnin eri toteutustekniikoita, kuten Java EE -sovelluspalvelintekniikkaa, OpenOfficen käyttöä sekä palvelimella että työasemassa, UNO-sovellusrajapinnan käyttöä ja XML-tekniikoiden käyttöä OpenDocument-muotoisten tiedostojen käsittelyssä.

Asiakirjojen käsittelyn automatisoinnissa on kuvattu ratkaisuja tietojen hakuun ja raportointiin, metatietojen keruuseen, asiakirjojen kokoamiseen ja niiden automaattisiin muunnoksiin. OpenDocument perustuu XML-rakenteeseen, ja asiakirjojen käsittelyn automatisointi voidaan siksi perustaa XSLT-pohjaisiin tuonti- ja vientisuotimiin. OpenOffice.org tukee OpenDocument-tiedostomuotoa, ja raportissa on annettu esimerkkejä XSLT-suotimien käytöstä OpenOfficessa sekä kuvattu OpenOfficen hallintaa ulkoisista ohjelmista UNO-komponenttimallin kautta. Raportissa on kuvattu myös erillisten ulkoisten XSLT-suotimien ja muiden ohjelmointivälineiden käyttöä OpenDocument-asiakirjojen käsittelyssä.

XForms on standardoitu XML-pohjainen merkintäkieli, joka on tarkoitettu tietolomakkeiden kenttien määrittelyyn ja kenttien yhdistämiseen XML-tietueisiin. XFormsin tärkeimpiä käyttökohteita ovat lomakkeet, joiden sisältö on mahdollista lähettää sähköisesti. OpenDocument-asiakirja voi sisältää XForms-lomakkeita, ja raportissa on kuvattu lyhyesti XFormsin käytön periaatteita.

Raportissa tarkastellaan myös toimisto-ohjelmiston toteutuksesta riippuvia ominaisuuksia, joita ei ole määritelty OpenDocument-tiedostomuotoon. Toteutuksesta tai käyttöympäristöstä riippuvat tekijät liittyvät esimerkiksi makroihin, tavutusalgoritmeihin ja käytössä oleviin fontteihin.

LÄHTEET

OpenDocument Wikipediassa:
http://en.wikipedia.org/wiki/Opendocument

OpenDocument-tiedostomuodon tekninen määritys Wikipediassa:
http://en.wikipedia.org/wiki/OpenDocument_technical_specification

OpenDocument-tiedostomuotoa tukevat ohjelmat Wikipediassa:
http://en.wikipedia.org/wiki/OpenDocument_software

OpenDocument-tiedostomuodon käyttöönottotietoja Wikipediassa:
http://en.wikipedia.org/wiki/OpenDocument_adoption

[Bel2006] **Belgian government embraces Open Document Format**. IDABC. 3.7.2006.
http://ec.europa.eu:80/idabc/en/document/5686

[Den2006] **Denmark taking the lead in governments IT policy**. IDABC. 14.6.2006.
http://ec.europa.eu/idabc/en/document/5646

[Eis2006] **OASIS OpenDocument Essentials—Using OASIS OpenDocument XML**.
David Eisenberg. 26.1.2006
http://books.evc-cit.info/

[Eu2004a] **Comparative assessment of Open Documents Formats - Market Overview**.
IDA/Valoris report ENTR/02/21-IDA/MIDDLEWARE-XML. Huhtikuu 2004.
http://europa.eu.int/idabc/servlets/Doc?id=1928

[Eu2004b] **TAC approval on conclusions and recommendations on open document formats**. IDABC. 25.5.2004.
http://europa.eu.int/idabc/en/document/2592/5588

[Eu2004c] **European Interoperability Framework for pan-European eGovernment Services. Documentation on the European Interoperability Framework.**
IDABC. Marraskuu 2004.
http://europa.eu.int/idabc/en/document/3473/5585#finalEIF

[Eu2006] **Conclusions and recommendations on Open Document Formats.**
PEGSCO (Pan-European eGovernment ServicesCommittee). 6.12.2006.
http://ec.europa.eu/idabc/servlets/Doc?id=26971

[Fin2006] **10 000 työasemaa avoimeen aikaan. Oikeusministeriö siirtyy Openofficeen.**
Tietokone. 5.12.2006.
http://www.tietokone.fi/uutta/uutinen.asp?news_id=29011

[Fin2007a] **Migrating a Ministry to OpenOffice.org**. Ministry of Justice, Operations and Administration 2007:2. ISBN 978-952-466-487-5. 19 February 2007.
http://www.om.fi/Etusivu/Julkaisut/Julkaisusarjat/Toimintajahallinto/Toiminnanjahallinnonarkisto/Toimintajahallinto2007/1171362109118

[Fin2007b] **Avointen formaattien tuki kuntalaisille. Uusikaupunki säästää Openofficella.** Tietokone. 28.3.2007.
http://www.tietokone.fi/uutta/uutinen.asp?news_id=30078

[Fre2006] **French administration opts for OpenOffice**. IDABC. 7.7.2006. http://ec.europa.eu:80/idabc/en/document/5695

[IBM2007] **What's new in IBM Lotus Notes and Domino V8**. IBM. 20.3.2007 http://www-128.ibm.com/developerworks/lotus/library/notes8-new/

[Iso2006] **ISO and IEC approve OpenDocument OASIS standard for data interoperability of office applications**. International Organization for Standardization. 8.5.2006. http://www.iso.org/iso/en/commcentre/pressreleases/2006/Ref1004.html

[Mas2005] **Commonwealth of Massachusetts. Enterprise Technical Reference Model - Version 3.5. Information Domain**. 21.9.2005. http://www.mass.gov/Aitd/docs/policies_standards/etrm3dot5/ETRM_v3dot5draft_information.pdf

[Odf2005] **OpenDocument v1.0 Specification. OASIS Open Document Format for Office Applications (OpenDocument)**. Organization for the Advancement of Structured Information Standards. 1.5.2005. http://www.oasis-open.org/committees/download.php/12572/OpenDocument-v1.0-os.pdf

Liite A. XML-merkintäkieli

XML on yleiskäyttöinen merkintäkieli, joka perustuu tekstimuotoisiin tiedostoihin. Monet tiedostomuodot, kuten OpenDocument tai XHTML, ovat tämän merkintäkielen sovelluksia. Sovellus määrittelee tarkasti, minkälaisia XML-merkintöjä tiedostossa voi olla. Merkinnät tehdään kulmasulkeilla merkityillä tunnisteilla, kuten `<henkilo>`. Yksinkertaisimmillaan XML-tiedosto on seuraavanlainen:

```
<?xml version="1.0" encoding="UTF-8"?>

<henkilo>
   Matti Meikäläinen
</henkilo>
```

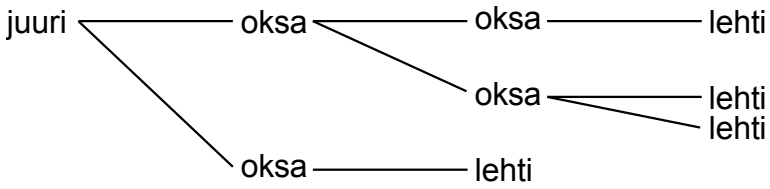
Ensimmäinen merkintä määrittelee, että kyseessä on XML-muotoinen tiedosto ja kertoo sen version ja merkkikoodauksen. Riveillä tai välilyöntien määrällä XML-tiedostossa ole mitään merkitystä, vain merkinnöillä. Yllä `<henkilo>`-merkintä aloittaa *elementin*, joka päättyy `</henkilo>`-merkintään. Täten avatun ja suljetun elementin sisällä voi olla tekstiä tai muita elementtejä. Lisäksi voi olla vastinparittomia elementtejä, jolloin merkintä päättyy `/`-merkkiin, esimerkiksi: `<siviilisaaty/>`. Erittäin tärkeää on, että merkinnät ovat aina sisäkkäisiä, eivätkä koskaan lomittaisia. Esimerkiksi seuraava on sallittua:

```
<juuri>
   <oksa>
      <oksa>
         <lehti/>
      </oksa>
      <oksa>
         <lehti/>
         <lehti/>
      </oksa>
   </oksa>
   <oksa>
      <lehti/>
   </oksa>
</juuri>
```

Seuraavanlainen lomittainen merkintä taas *ei* ole sallittua:

```
<juuri>
   <kappale>
      Tässä on normaalia tekstiä.
      <lihavoitu>Tämä on lihavoitua tekstiä
   </kappale>
   <kappale>
      tässä lihavointi jatkuu seuraavassa kappaleessa,</lihavoitu>
      kunnes päättyy.
   </kappale>
</juuri>
```

Sisäkkäisen hierarkian vaatimuksella on suuri merkitys kaikissa XML-tiedostomuodon käsittelyyn käytettävissä työkaluissa, mukaan lukien tässä raportissa mainitut XSLT ja XPath. Hierarkisuuden ansiosta XML-asiakirja voidaan esittää seuraavanlaisessa puumuodossa:



Kuva 18: XML-tiedoston esitys puuna.

Elementtien alkumerkinnöissä voi olla attribuutteja, jotka koostuvat attribuutin nimestä, yhtäsuuruusmerkistä ja lainausmerkkeihin kirjoitetusta arvosta, joka voi olla tekstiä tai numeerinen. Esimerkiksi:

```
<henkilo henkilotunnus="012345-0123A">
   <nimi>Matti Meikäläinen</nimi>
</henkilo>
```

Elementtien nimet voivat lisäksi sisältää merkinnän *nimiavaruudesta*. Nimiavaruuksia käytetään erottamaan elementit toisistaan, mikäli samassa tiedostossa voi olla elementtejä useasta XML-muodosta. Nimiavaruus mainitaan elementin alussa kaksoispisteellä erotettuna. Nimiavaruuden tunnisteet täytyy määritellä ylemmällä elementtitasolla yhdistämällä ne tiettyyn URI-osoitteeseen. Seuraavassa esimerkissä on käytetty omaa henkilötietojen esitykseen käytettävää XML-muotoa ja XHTML-muotoa samassa asiakirjassa:

```
<?xml version="1.0" encoding="UTF-8"?>

<henkilot xmlns:hlot="http://oma.uri.org/henkilot"
          xmlns:xhtml="http://www.w3.org/1999/xhtml">
   <hlot:henkilo hlot:henkilotunnus="012345-0123A">
      <hlot:nimi>Matti Meikäläinen</hlot:nimi>
      <hlot:kuvaus>
         <!-- Seuraavassa käytetään toista nimiavaruutta. -->
         <xhtml:p>
            Tässä on XHTML-muotoinen kappale, jossa kuvataan henkilöä.
            Tekstissä voi olla vaikka <xhtml:b>lihavointia</xhtml:b>.
         </xhtml:p>
      </hlot:kuvaus>
   </hlot:henkilo>
</henkilot>
```

Myös XSLT-kielessä käytetään nimiavaruuksia vahvasti, kuten voidaan nähdä monissa tämän raportin esimerkeissä, sekä liitteessä B. OpenDocument-tiedostomuodon nimiavaruuksista kerrotaan muun muassa luvussa 6.

XML-tiedostoissa voi olla myös kommentteja, jotka jätetään huomiotta XML-tiedostoa käsiteltäessä. Ne alkavat merkinnällä `<!--` ja päättyvät merkintään `-->`. Useimmissa tässä raportissa esitetyissä esimerkeissä on käytetty tällaisia kommentteja.

XML-tiedostomuotojen spesifikaatiot kirjoitetaan yleensä DTD- (Document Type Specification), Relax NG- tai muulla vastaavalla rakennemäärittelykielellä. Tällaista muodollista rakennemäärittelyä voidaan käyttää myös asiakirjojen kieliopillisen oikeellisuuden tarkistamisessa. DTD- ja Relax NG -rakennemäärittelyistä annetaan johdanto liitteissä D ja E.

Liite B. XSLT-muunnos

XML-muunnoskieli XSLT on tärkein OpenDocument-asiakirjojen muuntamiseen käytettävä ohjelmointikieli. Se saa syötteekseen XML-asiakirjan ja muuntaa sen toiseksi XML-asiakirjaksi tai vaihtoehtoisesti HTML-muotoon tai pelkäksi tekstiksi.

Esittelemme tässä liitteessä XSLT-kielen perusteet, sekä esittelemme joitain työkaluja, joilla voi tehdä XSLT-muunnoksia.

B.1. XML-tyylisivukieli XSL

XSLT on osa XSL-tyylisivukieltä (XML Stylesheet Language), joka koostuu kolmesta kielestä:

- **XPath**-osoituskieli
- **XSLT**-muunnoskieli
- **XSL-FO**-sivukuvauskieli

XPath on kieli, jonka avulla voidaan tehdä osoituksia XML-asiakirjassa. Sitä käytetään keskeisesti XSLT-muunnoksissa, mutta sitä voidaan käyttää myös muissa ohjelmointikielissä, sekä XML-tietokantakyselyissä kyselykielenä. Sitä käytetään muun muassa XForms-sidoksissa.

Näistä XML-FO ei ole oleellinen OpenDocumentin kannalta, sillä se määrittelee XML:n graafisen esityksen. XSL-FO on eräänlainen sivunkuvauskieli, jollaiseksi XML-asiakirja muunnetaan XSLT-muunnoksen avulla. XSL-FO voidaan edelleen muuntaa helposti erilaisiin tulostusmuotoihin, kuten näytölle, tulostimelle tai PDF:ksi.

B.2. XSLT-suorittimet

XSLT-suoritin on ohjelma, joka ottaa syötteekseen XSLT-ohjelman ja muunnettavan XML-asiakirjan. Suorittamalla XSLT-ohjelman se tuottaa tulosasiakirjan, joka on tyypillisesti XML-muotoista, mutta voi olla myös HTML-muotoista tai muotoilematonta tekstiä.

Useimmat XSLT-suorittimet ovat saatavilla ohjelmakirjastoina, joita käytetään monissa ohjelmissa, kuten Xalan-suoritinta OpenOfficessa. Alla on lyhyt luettelo tavallisimmin käytetyistä XSLT-suorittimista eri käyttöjärjestelmissä.

XSLT-suoritin	Rajapinnat	Huomautuksia
Xalan	C++ ja Java	OpenOfficen käyttämä. Osa Apache-projektia.
SAXON	Java ja .NET	Tekijä Michael Kay (XSLT Programmer's Reference)
Transformiix	C++	Mozilla Firefox -selaimen sisäinen suoritin
Gnome libxslt	C	Linux-käyttöjärjestelmän Gnome-projektin suoritin

Suorittimissa on jonkin verran eroja, lähinnä niiden tukemissa funktioissa. Kaikki toteuttavat XSLT:n pakolliset perusfunktiot, mutta eivät välttämättä kaikkia muita. Lisäksi useimmat suorittimet tarjoavat omia laajennusfunktioitaan. Kenties laajimmalle levinnyt laajennusfunktio on *node-set*(), jonka avulla muunnoksen tulospuusta voidaan muodostaa

lähdepuu, joka voidaan käsitellä uudelleen. Tämä sisältyy ainakin Xalan- ja SAXON-suorittimiin.

Suoritinkirjastoja käytetään monissa ohjelmissa. Osa näistä on komentoriviohjelmia, joiden avulla XSLT-muunnoksia voi tehdä sovellusohjelmista riippumatta. Alla on luettelo joistain tunnetuimmista.

Ohjelma	XSLT-suoritin	Huomautuksia
OpenOffice	Xalan	Muunnokset OpenDocument-muodosta ja muotoon
Mozilla Firefox	Transformiix	Monessa käyttöjärjestelmässä toimiva web-selain
xsltproc	Gnome libxslt	Komentoriviohjelma (Linux)
xalan	Xalan	Komentoriviohjelma (Linux, Windows, jne.)
msxsl.exe	Microsoft XSL Processor	Microsoftin komentoriviohjelma Windowsille. Vaatii Microsoft Core XML Services -ohjelmiston asennuksen.

Monissa tämän kirjan esimerkeissä komentorivillä tehtävistä muunnoksista käytetään xsltproc-ohjelmaa (Linux). Muunnoksen tulos tulostetaan oletustulosteeseen, eli normaalisti näytölle. Xsltprocia käytetään seuraavasti:

```
$ xsltproc muunnin.xsl muunnettava.xml
```

Toinen vaihtoehto on käyttää Xalan-suoritinta komentoriviltä (Linux, Windows, jne.). Xalan C++ -version mukana tulee Xalan-komentorivityökalu, jota käytetään seuraavasti:

```
$ xalan -in muunnettava.xml -xsl muunnin.xsl -out tulos.xml
```

Saman voi tehdä Xalanin Java-versiolla (Linux, Windows, jne.). Kun Java-ajoympäristö ja Xalan on asennettu asianmukaisesti, voidaan muunnos tehdä seuraavalla komennolla:

```
$ java org.apache.xalan.xslt.Process -IN syote.xml -XSL muunnos.xsl -OUT
tulos.xml
```

B.3. XPath-polkuosoitukset

XPath käsittelee XML-asiakirjaa puumaisena hierarkiana. XPath-polku koostuu luettelosta solmuja /-merkillä erotettuna. Polut ovat muutoin suhteellisia nykyisen tarkastelusolmun suhteen, mutta polun alussa annetulla /-merkillä ilmaistaan absoluuttinen polku juurisolmusta lähtien. Esimerkkejä:

- Suhteellinen polku: `henkilo/etunimi`
- Absoluuttinen polku: `/henkilot/henkilo/etunimi`

XSLT käyttää XPath-polkuja täsmäämällä ne XML-puuhun, jolloin tiettyyn polkuun voi täsmätä useita solmuja. Käytetään seuraavaa esimerkkiä, jossa on lisätty edellä käytettyyn henkilötietoesimerkkiin `<henkilo>`-elementin `id`-attribuutti:

```
<henkilot>
  <henkilo id="1">
    <etunimi>Aleksanteri</etunimi>
    <sukunimi>Suuri</sukunimi>
```

```
    <siviilisaaty tyyppi="naimaton"/>
  </henkilo>

  <henkilo id="2">
    <etunimi>Kaarle</etunimi>
    <sukunimi>Suuri</sukunimi>
    <siviilisaaty tyyppi="leski"/>
  </henkilo>
</henkilot>
```

Kaikkien henkilöiden etunimet saa haettua viittauksella `/henkilot/henkilo/etunimi`. Ilmaus siis täsmää kaikkiin etunimisolmuihin. Attribuutti saadaan täsmättyä @-merkillä.

XPath tukee vertailuoperaatioita ja loogisia operaatioita. Voimme esimerkiksi hakea kaikki Kaarle-nimiset henkilöt: `henkilo/[etunimi='Kaarle']`. Vastaavasti voimme verrata attribuuttia ja noutaa kaikki henkilöt, joiden `id`-tunniste on 2: `henkilo/[@id=2]`.

B.4. Hakulausekkeet

XSLT-muunnos poimii tietoja lähdetiedoston XML-puusta ja tuottaa tulostiedostoon erilaisia XML-elementtejä. Tietojen poimiminen tapahtuu kahdella tavalla: osoituksilla ja täsmäyshaulla.

XSLT:n tärkein elementti on `template`, jonka avulla voidaan hakea kaikki solmut, jotka täsmäävät `match`-attribuutissa annettuun XPath-lausekkeeseen. Muunnos tapahtuu siten, että ulostuloon kirjoitetaan `template`-lausekkeen sisältämä teksti, joka on tyypillisesti XML:ää, mutta voi olla myös HTML:ää tai vapaata tekstiä. Täsmäyksessä tarkastellaan ”nykyisen” solmun alasolmuja, aloittaen lähdeasiakirjan juurisolmusta.

Esimerkiksi seuraava lauseke täsmää juurisolmuun ja tuottaa ulostuloon XML-asiakirjan, jossa on listattuna kaikki henkilöt `<henkilo>...</henkilo>`-elementeissä ja niiden sisällä henkilöiden etunimet `<etunimi>...</etunimi>`-merkinnällä. Muunnoksen tulokseen tulevat elementit on alla esitetty kursiivilla.

```
<xsl:template match="henkilo">
    <etunimi>
        <xsl:value-of select="etunimi"/>
    </etunimi>
</xsl:template>
```

Elementti `xsl:value-of` palauttaa solmun tai alipuun tekstisisällön, tässä tapauksessa `<etunimi>...</etunimi>`-merkinnän sisällä olevan tekstin.

B.5. Nimiavaruudet

Nimiavaruudet ovat tekniikka, jolla pyritään estämään elementtien nimien ristiriitoja silloin, kun samassa XML-tiedostossa esiintyy usean eri XML-muodon elementtejä. Näin on juuri XSLT:ssä, Jos haluamme olla varmoja, että ristiriitaa ei synny, määrittelemme nimiavaruuden, joka tekee elementistä yksiselitteisen.

```
<jokuelementti
  xmlns:oma="http://fi.openoffice.org/2007/omamuoto">
  <oma:henkilot>
  </oma:henkilot>
</jokuelementti>
```

Nimiavaruuden varsinaiseksi nimeksi asetetaan yllä siis `http://fi.openoffice.org/2007/omamuoto` ja sen lyhenteeksi `oma`. Tämän jälkeen nimiavaruutta voidaan käyttää liittämällä lyhenne elementin eteen kaksoispisteellä erotettuna.

Koska OpenDocumentia käsittelevä XSLT-ohjelma sisältää yleensä myös OpenDocument-muodon elementtejä, täytyy niiden nimiavaruudet määritellä `xsl:stylesheet`-elementissä. Huomaa, että XSLT-elementtien nimiavaruuden lyhenne on `xsl`.

```
<xsl:stylesheet
  xmlns:omamuoto="http://fi.openoffice.org/2007/omamuoto"
  version="1.0">

</xsl:stylesheet>
```

B.6. Merkkikoodauksen asettaminen

Eräs XML:n tärkeimmistä yksityiskohdista on, että jokaisella asiakirjalla on määriteltynä sen käyttämä merkkikoodaus. Se määritellään kaikkien XML-tiedostojen ensimmäisellä rivillä, tyypillisesti seuraavasti:

```
<?xml version="1.0" encoding="UTF-8"?>
```

UTF-8 on XML-tiedostojen oletusmerkkikoodaus, sillä se kykenee esittämään kaikki Unicode-standardin kymmenet tuhannet merkit. Yksitavuiset ISO 8859-1 (länsimaalainen) ja ISO 8859-15 (länsimaalainen euromerkillä) -koodaukset ovat silti edelleen hyvin käytettyjä, erityisesti koska useimmat tekstieditorit tukevat niitä hyvin. Tämänkin oppaan käsin kirjoitetuissa XML-tiedostoissa käytetään siksi otsikkoa:

```
<?xml version="1.0" encoding="ISO-8859-15"?>
```

XSLT-muunnoksen tulostiedoston merkkikoodaus on myös oletuksena UTF-8. Se voidaan asettaa joksikin toiseksi.

B.7. XSLT:n rajat

Jotkin muunnokset ovat kuitenkin vaikeita tehdä XSLT:llä, jolloin täytyy turvautua muihin ohjelmointikieliin. Esimerkkinä erittäin vaikeasta muunnoksesta on SVG-vektorigrafiikka, jonka XML sisältää ohjelmakoodia, jota täytyy pystyä suorittamaan. Ulostulomuodon saatettaisiin haluta olevan pisterasterikuva, vieläpä pakattu sellainen. XSLT:ssä ei kuitenkaan ole muuttujia, joiden avulla kuva voitaisiin esittää käsittelyn aikana. Yksinkertainen merkkijonokäsittelykin on sillä melko vaivalloista. XSLT ei yksinkertaisesti sovellu tämäntyyppisiin tehtäviin.

OpenDocument-asiakirjoissa olevia, XSLT:llä vaikeasti muunnettavia asioita voisivat olla esimerkiksi seuraavat:

- Taulukkolaskennan funktiot
- Basic-makrot
- Impress-animaatiot

Liite C. Identiteettimuunnos

OpenDocument-asiakirja voidaan esittää kahdella tavalla: joko useana XML-tiedostona, jotka on pakattu ZIP-tiedostoksi, tai yhtenä XML-tiedostona. Oletuksena käytetään usean XML-tiedoston ZIP-pakettia, eikä OpenOfficessa edes ole vientisuodatinta, jolla asiakirjan voisi tallentaa yhteen XML-tiedostoon. Yhden tiedoston XML-esitysmuotoon törmää suodattimissa, sillä vientisuodatin saa asiakirjan syötteekseen yhtenä XML-tiedostona. Samoin tuontisuodatin voi tuottaa koko asiakirjasisällön tulostamalla sen yhtenä XML-tiedostona.

Viennin yhden XML-tiedoston muotoon saa tehtyä helposti identiteettimuunnoksella, joka tuottaa täydellisen kopion XML-puusta. Tallennetaan seuraava muunnos nimellä `identity.xsl`:

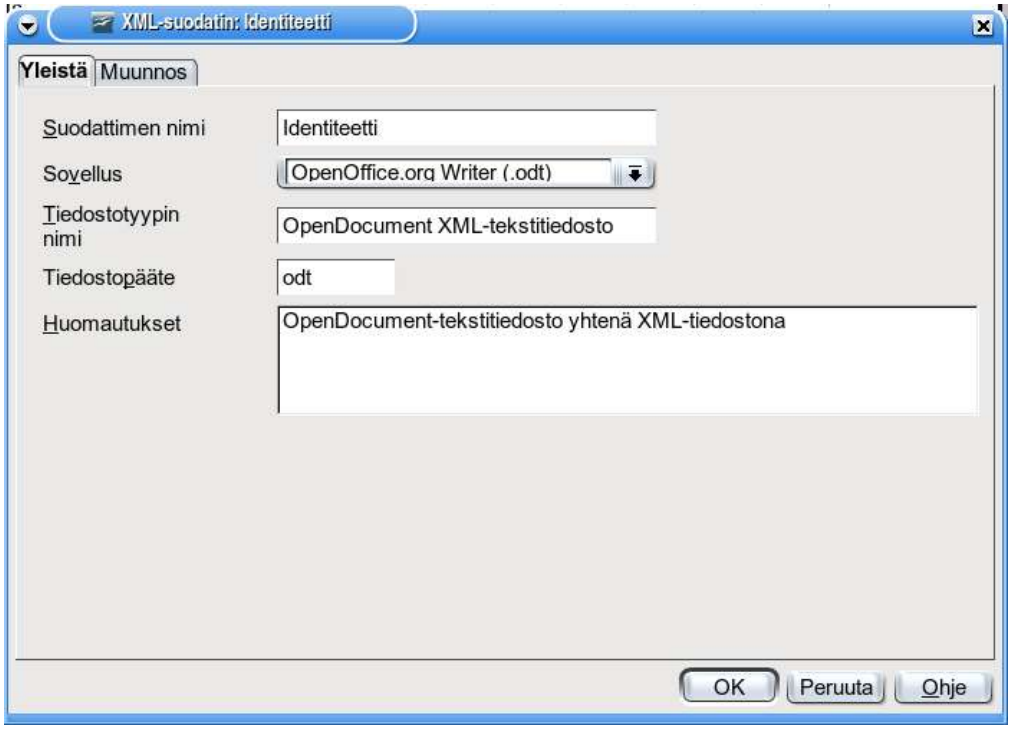
```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                version="1.0">

  <xsl:output indent="yes"/>

  <xsl:template match="@*|node()">
    <xsl:copy>
      <xsl:apply-templates select="@*|node()"/>
    </xsl:copy>
  </xsl:template>

</xsl:stylesheet>
```

Jotta tulos-XML olisi helppolukuisempaa, määrätään `<xsl:output indent="yes"/>`-merkinnällä, että elementtien loppumerkinnän jälkeen tehdään rivinvaihto. Mikäli asiakirjassa on binäärimuotoista dataa, kuten kuvia, koodataan ne binääridatakoodauksella.

Yhden XML-tiedoston muodon käsittely on monelta osin hankalampaa kuin ZIP-tiedoston. Erityisesti binäärimuodossa olevat kuvat ovat ongelmallisia, sillä niiden koodaus XML:ssä on erityisen tehotonta. Kuvia ei ole mahdollista käsitellä XSLT:llä mitenkään. Kuitenkin niiden lukeminen XSLT-suorittimeen vie paljon aikaa ja erityisesti muistia.

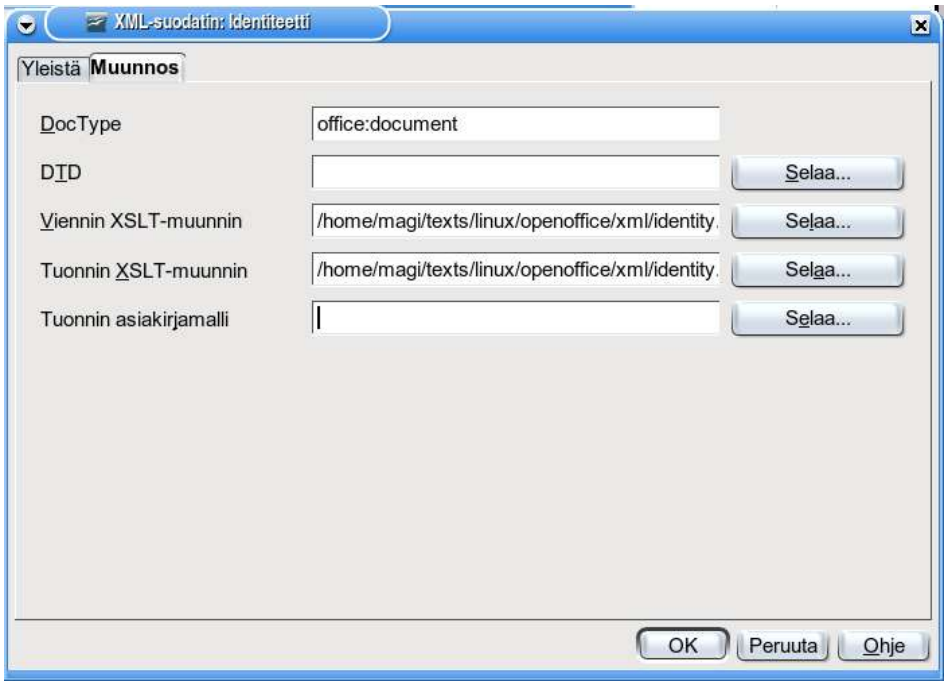
C.1. Identiteettisuodattimen asentaminen

Luodaan uusi vientisuodatin valikosta **Työkalut > XML-suodattimien asetukset**. Napsauta **Uusi** ja tee valintaikkunaan seuraavat asetukset:



Sovellus-asetus määrää, missä sovelluksessa asiakirja avataan. Tässä teemme Writer-sovelluksessa avattavan tekstiasiakirjan. **Tiedostotyypin nimi** on nimi, jonka voi valita tiedoston avaus- ja tallennusikkunassa, kun halutaan avata tai tallentaa tiedosto määriteltävää suodatinta käyttäen. **Tiedostopääte** määrää tallennuksessa käytettävän tiedostopäätteen. Käytämme tässä samaa päätettä `.odt` kuin ZIP-paketissakin olevassa tallennusmuodossa, mutta selkeyden vuoksi voi käyttää myös muuta päätettä.

Tee **Muunnos**-välilehden asetukset seuraavasti:



DocType-asetus auttaa OpenOfficea tunnistamaan XML-tiedoston OpenDocument-asiakirjaksi sen ylimmän tason elementin perusteella. **DTD**-asetusta ei tässä tarvita. Sekä

Viennin XSLT-muunnin että **Tuonnin XSLT-muunnin** asetetaan samaksi `identity.xsl`-muuntimeksi. Koska yhden XML-tiedoston muoto sisältää tyylimääritykset, ei **Tuonnin asiakirjamalli** -määritystä tarvita.

C.2. Identiteettisuodattimen käyttö

Luo uusi tekstiasiakirja ja kirjoita siihen jotain sisältöä. Tallenna tiedosto valikosta **Tiedosto > Tallenna nimellä**. Valitse tallennusikkunassa **Tiedoston tyyppi** -kohdasta määrittelemäsi tiedostomuoto, joka on luettelossa muodossa **OpenDocument XML-tekstitiedosto (.odt)**. Tallennettaessa ohjelma saattaa varoittaa tallentamisesta muuhun kuin OpenDocument-muotoon. Ohita varoitus valitsemalla **Kyllä**. Tallennuksen jälkeen voit sulkea asiakirjan.

Tiedoston avaaminen tapahtuu kuten minkä tahansa tiedoston, valikosta **Tiedosto > Avaa**. Koska määrittelimme tiedoston päätteeksi `.odt` ja DocType-määritykseksi `office:document`, tunnistaa OpenOffice tiedoston määrittelemämme suodattimen muotoiseksi ja avaa sen sitä käyttäen. Mikäli haluat, voit myös valita tyypin manuaalisesti **Tiedoston tyyppi** -valinnasta.

Liite D. DTD-rakennemäärittelyt

DTD on laajalti käytetty SGML- ja XML-rakennemäärittelykieli. OpenOfficessa sitä voidaan käyttää tuontisuotimissa, tuotavan asiakirjan oikeellisuuden tarkistamiseen. DTD on suunniteltu vanhan XML 1.0:n tarpeita varten ja sitä tuetaan erittäin laajalti. Uudemmat XML Schema ja RelaxNG-määrittelykielet ovat kuitenkin vähitellen syrjäyttäneet DTD:tä, koska se on riittämätön uudemman XML 1.1:n tarpeisiin. Se ei muun muassa tue nimiavaruuksia, eikä muutenkaan ole riittävän ilmaisukykyinen kaikkien XML-piirteiden kuvaamiseen.

DTD-määrittely koostuu ylimmällä tasolla ELEMENT-määrityksistä, joilla määritellään XML-asiakirjassa esiintyvät elementit, kuten `<henkilo></henkilo>`. Yllä mainitun henkilötietoesimerkin DTD-rakennemäärittely olisi seuraavanlainen:

```
<!ELEMENT henkilot (henkilo*)>
<!ELEMENT henkilo (etunimi,sukunimi,syntymaaika,ammatti,siviilisaaty)>
<!ELEMENT etunimi (#PCDATA)>
<!ELEMENT sukunimi (#PCDATA)>
<!ELEMENT syntymaaika (#PCDATA)>
<!ELEMENT ammatti (#PCDATA)>
<!ELEMENT siviilisaaty EMPTY>
<!ATTLIST siviilisaaty tyyppi (naimaton|naimisissa|eronnut|leski) #REQUIRED>
```

Elementin `ELEMENT`-määrityksessä kerrotaan, mitä muita elementtejä elementti voi sisältää ja montako. Asteriski elementin perässä, esimerkiksi `henkilo*`, määrää että henkilö-elementtejä saa olla vapaa määrä tai ei yhtään. Plusmerkki + määrää, että elementtejä täytyy olla ainakin yksi. Jos kumpaakaan määrettä ei ole, täytyy elementtejä olla tasan yksi. Esimerkiksi yllä `etunimi`-elementtejä tulee kullakin `henkilo`-elementillä olla tasan yksi. Arvo `#PCDATA` kertoo, että elementillä saa olla tekstisisältöä, kuten `<etunimi>Aleksanteri</etunimi>`. Jos arvo on EMPTY, ei elementillä saa olla lainkaan tekstisisältöä, vaan sen täytyy olla muotoa `<elementti/>`.

Elementin attribuutit määritellään `ATTLIST`-määrityksellä. Sen parametrejä ovat elementti, attribuutti ja attribuutin mahdolliset arvot. Yllä `siviilisaaty`-elementin `tyyppi`-attribuutilla voi olla neljä arvoa. Sallittu sisältö on esimerkiksi: `<siviilisaaty tyyppi="naimaton">`.

Seuraavassa esimerkki käyttäen `onsgmls`-validaattoria, kun XML-tiedostoon on tehty virhe:

```
$ onsgmls -wxml -s henkilotiedot.xml
onsgmls:henkilotiedot.xml:29:13:E: element "EETUNIMI" undefined
onsgmls:henkilotiedot.xml:30:13:E: document type does not allow element
"SUKUNIMI" here
onsgmls:henkilotiedot.xml:31:16:E: document type does not allow element
"SYNTYMAAIKA" here
onsgmls:henkilotiedot.xml:32:12:E: document type does not allow element
"AMMATTI" here
onsgmls:henkilotiedot.xml:33:34:E: document type does not allow element
"SIVIILISAATY" here
onsgmls:henkilotiedot.xml:34:11:E: end tag for "HENKILO" which is not
finished
```

Liite E. Relax NG -rakennemäärittelyt

XML-pohjainen Relax NG on DTD-määrittelyjä uudempi kieli XML-rakennemäärittelyjen tekemiseen ja mahdollistaa mm. tietotyyppien (kuten merkkijono tai kokonaisluku) määrittämisen. OpenDocument-tiedostomuodon rakennemäärittely on tehty Relax NG -kielellä.

Seuraavassa taulukossa on lueteltu joitain Relax NG:n tavallisimpia elementtejä.

`<element name="...">`	Määrittelee elementin, jonka nimi annetaan `name`-attribuutissa.
`<attribute name="...">`	Määrittelee attribuutin, jonka nimi annetaan `name`-attribuutissa.
`<text/>`	`<element>`-elementin sisällä määrittelee, että elementin sisällä voi olla tekstiä.

`<choise>`
`  <value>...</value>`
`  <value>...</value>`
`  ...`
`</choise>`
	`<attribute>`-elementin sisällä määrittelee attribuutille sallitut arvot.

`<zeroOrMore>`	Määrittelee, että sisällytettyjä elementtejä voi olla mielivaltainen määrä tai ei lainkaan.

Esimerkiksi sivulla 47 esitetyn henkilötietoesimerkin XML-asiakirjatyypin Relax NG -rakennemäärittely olisi seuraavanlainen:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<element name="henkilot" xmlns="http://relaxng.org/ns/structure/1.0">
  <zeroOrMore>
    <element name="henkilo">
      <element name="etunimi">
        <text/>
      </element>
      <element name="sukunimi">
        <text/>
      </element>
      <element name="syntymaaika">
        <text/>
      </element>
      <element name="ammatti">
        <text/>
      </element>
      <element name="siviilisaaty">
        <attribute name="tyyppi">
          <choise>
            <value>naimaton</value>
            <value>naimisissa</value>
            <value>eronnut</value>
            <value>leski</value>
          </choise>
        </attribute>
      </element>
    </element>
  </zeroOrMore>
</element>
```

Relax NG -rakennemäärittelyjä voidaan käyttää erityisesti oikeellisuustarkistuksiin. Seuraavassa esimerkki käyttäen `jing`-tarkistinta, kun XML-tiedostoon on tehty virhe:

```
$ jing henkilotiedot.rng henkilotiedot.xml
/home/magi/texts/linux/openoffice/xml/henkilotiedot.xml:29: error: unknown
element "eetunimi"
/home/magi/texts/linux/openoffice/xml/henkilotiedot.xml:30: error: required
elements missing
```